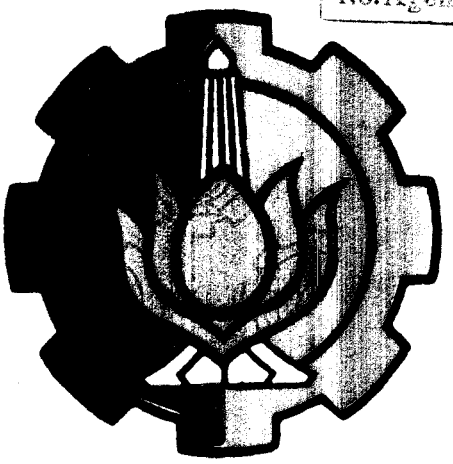
ALAT PENGIRIM DATA DIGITAL MELALUI PEMANCAR RADIO FM – STEREO YANG DAPAT DIINTERFACEKAN DENGAN IBM – PC



Oleh :

EDWIN ARISTIAWAN

NRP. 2882200941

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1993

ALAT PENGIRIM DATA DIGITAL MELALUI PEMANCAR RADIO FM – STEREO YANG DAPAT DIINTERFACEKAN DENGAN IBM – PC

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Teknik Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. MOCH. MOEFADOL ASYARI)	(Ir. HENDRA KUSUMA)

S U R A B A Y A
AGUSTUS, 1993

ABSTRAK

Perkembangan sistem komunikasi data antar komputer dewasa ini memungkinkan penggunaan media udara untuk jalur komunikasinya sebagai pengganti media kabel yang selama ini banyak digunakan.

Tugas Akhir ini membahas kemungkinan penggunaan media udara sebagai jalur komunikasi untuk sistem komunikasi data. Sebagai penghubung antara terminal komputer dan udara digunakan pemancar radio FM-Stereo, dimana data digital dari komputer terlebih dahulu dikonversikan menjadi besaran frekuensi dengan menggunakan metode Frequency Shift Keying.

Pembuatan alat komunikasi data ini dimaksudkan untuk lebih mengoptimalkan fungsi dari pemancar radio FM-Stereo yang telah memasyarakat dan sebagai media alternatif pada sistem komunikasi data di masa yang akan datang.

KATA PENGANTAR

Dengan mengucap puji syukur ke hadirat Allah SWT atas segala berkat dan rahmat-Nya, akhirnya penulis dapat menyelesaikan tugas akhir yang berjudul :

ALAT PENGIRIM DATA DIGITAL

MELALUI PEMANCAR RADIO FM-STEREO

YANG DAPAT DIINTERFACEKAN DENGAN IBM PC

Tugas akhir ini disusun guna memenuhi persyaratan untuk memperoleh gelar sarjana pada Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya.

Pada kesempatan ini pula penulis menyampaikan ucapan terima kasih dan penghargaan yang sebesar-besarnya dan setulusnya kepada :

- Bapak Ir. Mochammad Moefadol Asyari, selaku dosen pembimbing Tugas Akhir.
- Bapak Ir. Hendra Kusuma, selaku dosen wali dan dosen pembimbing Tugas Akhir.
- Bapak Ir. Soetikno, selaku Koordinator Bidang Studi Elektronika Jurusan Teknik Elektro FTI - ITS.
- Bapak Ir. Katjuk Astrowulan, MSEE; selaku Ketua Jurusan Teknik Elektro FTI - ITS.

- Rekan - rekan penulis dan semua pihak yang telah banyak membantu memberikan sumbangan pikiran, sehingga tugas akhir ini dapat terselesaikan.

Penulis menyadari adanya kekurangan-kekurangan dalam penyelesaian tugas akhir ini. Oleh karena itu penulis dengan segala kerendahan hati mengharapkan kritik dan saran demi kesempurnaan tugas akhir ini.

Besar harapan penulis semoga tugas akhir ini dapat memberikan sumbangan pengetahuan bagi siapa saja yang memanfaatkannya.

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Era teknologi komputer telah mendapat perhatian yang sangat besar dari banyak orang di dunia ini, sehingga komuputer sudah memasuki hampir setiap kehidupan manusia . Teknologi komputer berevolusi dengan cepatnya di segala bidang, mulai dari pekerjaan yang sederhana sampai proyek-proyek besar yang kompleks. Salah satu dampak perkembangan ini ialah timbulnya kebutuhan baru bagi pemakai-pemakai komputer tersebut, yaitu kebutuhan untuk saling bertukar informasi antar komputer, yang dapat dilakukan dengan menghubungkan satu komputer dengan komputer lain, baik melalui media kabel atau melalui media udara.

1.1 LATAR BELAKANG

Hubungan antar komputer dewasa ini pada umumnya dilakukan dengan menggunakan kabel sebagai media penghantarnya. Hal ini disebabkan karena media tersebut

mempunyai gangguan (*noise*) yang relatif lebih kecil dibandingkan bila menggunakan media penghantar lainnya. Sehingga data yang diperoleh mempunyai akurasi yang tinggi. Untuk jarak jauh, telepon paling banyak dipakai sebagai media perantara antara komputer sebagai pengolah dan kabel sebagai media penghantar data. Telepon paling banyak dipakai sebagai perantara karena mempunyai keuntungan-keuntungan di antaranya :

1. Praktis dalam penggunaannya, dengan hanya menghubungkan komputer dengan modem, selanjutnya dapat dihubungi nomor telepon tujuan untuk keperluan pertukaran data
2. Luas jangkauannya, dengan telepon dapat dihubungi berbagai tempat di dunia yang dikehendaki dengan mudah dan cepat; dengan syarat daerah yang akan dihubungi tersebut telah dimasuki jaringan telepon.

Selain dari keuntungan-keuntungan di atas, telepon juga mempunyai kekurangan-kekurangan, yaitu :

1. Biaya yang mahal, kebanyakan semua aktivitas pertukaran data membutuhkan waktu yang relatif lama sehingga biaya yang dikeluarkan untuk keperluan tersebut relatif mahal
2. Kemungkinan berhasilnya suatu sambungan (*dialing*) pada saat ini masih belum 100%
3. Pada saat ini belum seluruh wilayah Indonesia yang

terjangkau jaringan telepon.

Dengan pertimbangan berbagai hal di atas, maka penyusun mengambil judul :

ALAT PENGIRIM DATA DIGITAL MELALUI PEMANCAR RADIO FM-STEREO YANG DAPAT DIINTERFACEKAN DENGAN IBM-PC

sebagai perwujudan usaha-usaha untuk mencari alternatif media perantara yang lain selain media telepon.

1.2 PERMASALAHAN

Pengembangan akan dilakukan untuk mencari alternatif lain dalam sistem komunikasi data yang ada tersebut dihadapkan pada beberapa masalah, yang pertama adalah bagaimana data informasi yang dikirim pada pemancar dapat diterima pada penerima dengan kesalahan yang relatif kecil. Yang kedua, adalah cara pengolahan data yang dilakukan pada bagian pemancar dan bagian penerima. Untuk itu perlu dikaji bagaimana sistem komunikasi dan sistem pengolahan data yang efektif untuk mendukung keperluan ini.

1.3 PEMBATASAN MASALAH

Pembatasan pada tugas akhir ini adalah untuk mengirimkan data digital untuk spesifikasi karakter dan disertai dengan proses pengolahan datanya. Selain itu juga mengusahakan agar pada bagian penerima dapat

mereproduksi data digital yang dikirim pada bagian pemancar dengan kesalahan yang relatif kecil.

1.4 TUJUAN

Hasil akhir dari Tugas Akhir ini merupakan perangkat lunak dan perangkat keras guna mewujudkan rangkaian pengirim data digital melalui pemancar radio FM-Stereo pada hubungan antar komputer.

Dengan demikian diharapkan bahwa alat ini dapat merupakan alternatif lain dalam hal komunikasi data antar komputer di masa yang akan datang.

1.5 METODOLOGI

Untuk mencapai tujuan yang diinginkan, maka dalam pengerjaan tugas akhir ini dilakukan langkah sebagai berikut :

Studi literatur mengenai transmisi data serial, interface serial RS-232-C dan pin-pin DB-25, teknik-teknik modulasi FM-Stereo, teknik pengkodean data digital ke analog dan sebaliknya.

Selanjutnya dilakukan studi mengenai komponen yang dapat memenuhi kriteria-kriteria yang telah ditentukan. Kriteria ini bisa mengalami perubahan disesuaikan dengan karakteristik komponen yang ada.

Langkah berikutnya adalah disain rangkaian,

dimulai dari blok diagram, kemudian rangkaian lengkap dan disain PCB. Berikut perangkat lunak pendukung perangkat keras.

Langkah terakhir adalah uji coba pengoperasian peralatan dari sistem yang dibuat.

1.6 SISTEMATIKA PEMBAHASAN

Buku laporan Tugas Akhir ini disusun dengan sistematika sebagai berikut :

Pada BAB I - Pendahuluan, dijelaskan mengenai latar belakang pemilihan judul Tugas Akhir, permasalahan, pembatasan masalah, tujuan dibuatnya peralatan, metodologi serta sistematika pembahasan, dan relevansi pembuatan Tugas Akhir ini.

Pada BAB II - Teori Penunjang, dijelaskan mengenai mikroprosesor 8088, mikroprosesor 80286, transmisi data, interface serial RS-232-C, teknik pengkodean data digital ke analog, dan teknik modulasi FM-Stereo.

Pada BAB III - Perencanaan dan Pembuatan, dibahas mengenai konsep perancangan peralatan yang dibuat berikut perangkat lunak pendukungnya.

Pada BAB IV - Pengujian Peralatan, dibahas mengenai pengujian pengoperasian peralatan.

Pada BAB V - Kesimpulan, dibahas mengenai

kesimpulan seluruh pembahasan, peralatan yang dibuat serta saran-saran untuk pengembangan peralatan lebih lanjut.

1.7 RELEVANSI

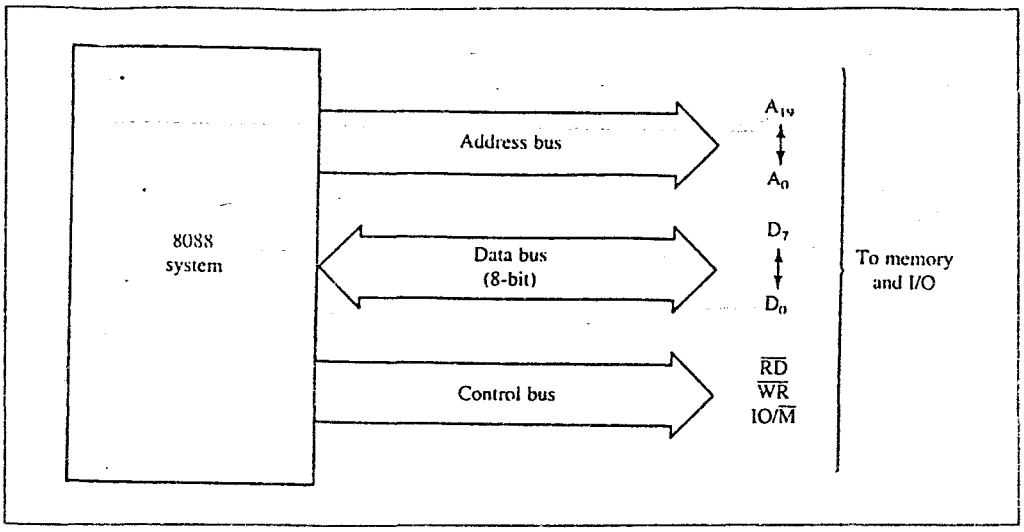
Dari pembuatan tugas akhir ini diharapkan diperoleh metode perancangan sistem yang tepat dan alat yang dibuat dapat digunakan untuk melengkapi sistem pengiriman data digital yang ada pada saat ini, sekaligus sebagai alternatif lain dalam hal komunikasi data. Diharapkan sistem ini dapat dikembangkan untuk masa-masa yang akan datang.

BAB II
TEORI PENUNJANG

2.1 SISTEM MIKROPROSESOR

Sebuah mikroprosesor adalah merupakan produk dari teknologi Large Scale Integration (LSI) yang menghasilkan implementasi CPU dalam satu serpih. Dan sebuah Sistem Mikroprosesor adalah implementasi dari mikroprosesor (CPU) atau lebih dengan ditunjang oleh beberapa unit fungsional yang lain. Sebuah Sistem Mikroprosesor yang sederhana terdiri atas :

- Unit pengolah pusat (CPU, Central Processing Unit) dalam sebuah chip tunggal yang di dalamnya terdapat ALU (Arithmetic Logical Unit) yang melakukan perhitungan dan unit pengendali yang berfungsi untuk menyerempakkan operasi sistem serta register-register internal.
- Unit Memory yang berfungsi untuk menyimpan informasi.
- Unit Input-Output (I/O) yang berfungsi untuk komunikasi dengan dunia luar.



GAMBAR 2-1[1)]

SISTEM MIKROPROSESOR

Antara unit-unit di atas direalisasikan dalam arsitektur sistem 3 (tiga) bus yaitu:

-Address Bus

-Data Bus

-Control Bus

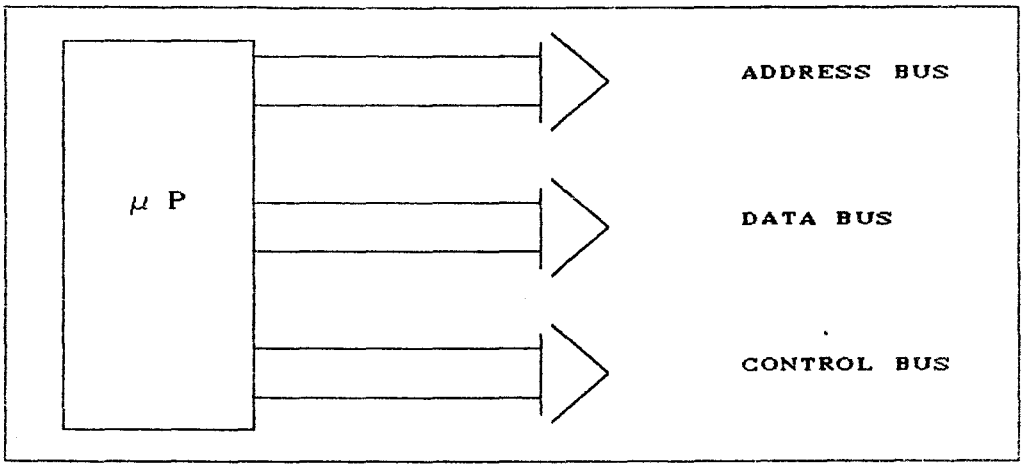
Sistem mikroprosesor standar dengan arsitektur sistem 3 (tiga) bus ditunjukkan pada gambar 2-1.

2.1.1 ARSITEKTUR SISTEM 3 BUS

Sistem bus didefinisikan sebagai kumpulan dari sinyal-sinyal komunikasi yang dikelompokkan berdasarkan fungsinya, di mana masing-masing mempunyai fungsi sebagai penghubung dalam suatu sistem.

---

1) Brey, B.B., THE INTEL MICROPROCESSORS, Macmillan Publishing Company, 1991, hal. 6



GAMBAR 2-2

ARSITEKTUR SISTEM 3 BUS

Sinyal-sinyal yang terdapat pada mikroprosesor 8 (delapan) bit dapat digolongkan menjadi 3 (tiga) buah sistem bus, yaitu :[2)]

1. Sistem Address Bus
2. Sistem Data Bus
3. Sistem Control Bus

Karena terdiri dari 3 (tiga) buah sistem bus maka disebut sebagai arsitektur sistem 3 (tiga) bus. Blok diagram sistem mikroprosesor dengan arsitektur 3 (tiga) bus dapat dilihat pada gambar 2-2.

2.1.1.1 SISTEM ADDRESS BUS

Address Bus digunakan untuk mentransmisikan

[2)] Coffron, J.W., PRACTICAL HARDWARE DETAIL FOR 8080, 8085, Z80, AMD 6800 MICROPROCESSOR SYSTEM, New Jersey, Prentice Hall Inc., 1981, hal. 3

alamat ke dalam lokasi memori atau piranti I/O yang memungkinkan pentransferan data. Sistem address bus merupakan output dari mikroprosesor yang akan membawa alamat-alamat menuju ke semua alat yang dihubungkan dengan data bus. Karena address bus merupakan pin output, maka hanya mempunyai satu arah saja (*Unidirectional*). Pada mikroprosesor 8 (delapan) bit seperti mikroprosesor 8088, bus alamat adalah 20 bit, yang memungkinkan pengiriman sampai 1024 Kbyte (2^{20}) lokasi alamat ekstern.

Pada 8 (delapan) bit address yang mempunyai orde rendah digunakan untuk menentukan lokasi I/O, maka untuk I/O hanya dapat mempunyai lokasi sebanyak 2^8 = 256 lokasi. Bus alamat digunakan dalam hubungannya dengan bus data untuk menentukan sumber atau tujuan data yang dikirim pada bus data.

2.1.1.2 SISTEM DATA BUS

Bus data berfungsi untuk mengirim dan menerima data antara berbagai serpih yang terdapat dalam sistem mikroprosesor. Jadi data bus bersifat dua arah (*Bidirectional*).

Akan tetapi walaupun data bus bersifat bidirectional, dalam waktu yang bersamaan data bus tidak dapat digunakan untuk mengirim dan menerima data. Jadi

sinyal hanya dapat dikirim atau diterima pada satu saat.

CPU 8088 dilengkapi dengan 8 (delapan) data biner yang diberi simbol D0 - D7, di mana D0 adalah Least Significant Bit (LSB) sedang D7 adalah Most Significant Bit (MSB). Data bus secara serentak membawa data valid dalam 8 (delapan) bit atau data 1 byte, dan oleh karena itu CPU 8088 disebut mikroprosesor 8 (delapan) bit.

2.1.1.3 SISTEM CONTROL BUS

Sistem control bus pada mikroprosesor mempunyai 4 (empat) buah sinyal yang mempunyai fungsi yang sama, yaitu sebagai kontrol pada mikroprosesor. Keempat sinyal tersebut adalah :

1. Memory Read (MEMR)
2. Memory Write (MEMW)
3. I/O Read (IOR)
4. I/O Write (IOW)

Kontrol bus ini merupakan sinyal output yang bersifat satu arah (*Unidirectional*). Sinyal MEMR aktif menunjukkan bahwa mikroprosesor sedang membaca data dari memori, bila sinyal MEMW aktif menunjukkan bahwa mikroprosesor sedang menulis data ke memori, sedang untuk IOR dan IOW menyatakan bahwa mikroprosesor sedang menerima atau mengirim data ke peralatan luar. Dengan demikian fungsi sinyal kontrol adalah untuk menentukan

hubungan kerja antara mikroprosesor dengan memori atau I/O, baik pada kondisi Read atau Write.

2.1.2 ADDRESS DECODING

Decoding berfungsi untuk mendekodekan saluran address CPU untuk mengaktifkan komponen memori maupun komponen I/O agar data yang akan dibaca atau ditulis pada piranti tersebut tidak kacau. Hal ini dapat dimungkinkan dengan suatu cara yang disebut dengan Address Decoding, di mana sinyal-sinyal pada address bus di-dekode sedemikian rupa sehingga setiap kombinasi pada address akan menghasilkan satu sinyal pilih yang akan mengaktifkan salah satu IC. Untuk dapat berkomunikasi dengan peralatan I/O diperlukan suatu cara agar mikroprosesor dapat memilih atau menentukan salah satu IC atau peralatan tertentu saja dengan mengaktifkan Chip Select (CS).

Beberapa metode yang dipergunakan untuk mendekode suatu address, yaitu dengan metode Gating dan Decoding.

- Gating : merupakan cara yang paling sederhana untuk mendekode suatu kombinasi address dengan menggunakan gate-gate (gerbang-gerbang) logika seperti AND, OR, dan NOT. Cara ini tidak banyak digunakan karena tidak efisien untuk sinyal pilih yang banyak.

- **Decoding** : Cara yang paling mudah dan efisien untuk decoding adalah dengan menggunakan dekoder. Dengan menggunakan dekoder maka setiap n macam kombinasi address dapat diperoleh sebanyak 2^n buah sinyal pilih, sehingga dekoder ini cocok untuk sistem yang memerlukan sinyal pilih yang banyak.

2.1.3 BUFFERING

Tiap masukan sebuah alat merupakan beban pada saluran yang menggerakkannya. Sebagian besar komponen menggerakkan komponen mulai dari satu sampai dua puluh komponen lainnya. Setiap komponen harus diperiksa karakteristik penggerakan serta pembebanan keluarannya.

Bus mikroprosesor harus berhubungan dengan setiap serpih masukan dan keluaran periferal dan memori sistem. Semua mikroprosesor MOS kurang mempunyai kemampuan penggerakan keluarannya dalam suatu sistem besar. Jika miroprosesor dibebani melebihi fan out, akibatnya level tegangan pada pin-pin bus dapat turun sampai melampaui batas level marginnya, yang dapat mengakibatkan mikroprosesor memberikan informasi yang tidak benar, karena itu dipakai buffer atau penggerak untuk menaikkan daya penggerakan bus.

Sebagai suatu contoh kita misalkan jalur address

pada mikroprosesor harus mendrive input address dari empat komponen memori dan dua komponen I/O, maka besarnya arus supply untuk mendrive komponen memori dan komponen I/O harus sama dengan jumlah arus input dari keempat komponen memori ditambah dengan jumlah arus input dari kedua komponen I/O.

2.1.4 MEMORY

Memory meruupakan tempat penyimpanan informasi yang dapat berupa data atau instruksi bagi suatu sistem mikroprosesor. Memory menurut fungsinya terdiri dari 2 (dua) jenis, yaitu : Read Only Memory (ROM) dan Random Access Memory (RAM).

2.1.4.1 READ ONLY MEMORY

Dalam sistem mikroprosesor diperlukan suatu memory untuk menyimpan data yang sulit untuk dihapus (*non volatile*), di mana data yang terdapat pada memory tersebut tidak akan hilang bila tegangan supply dari sistem dimatikan. Pada waktu isi memory ini ditentukan oleh proses rakitan, maka isinya tidak dapat lagi diubah. Isinya dapat dibaca, tetapi tidak dapat diisi lagi dengan program baru. Karena isi dari ROM tidak mudah dihapus (*non volatile*), maka ROM dipakai untuk menyimpan program-program utama.

Terdapat beberapa jenis memory yang bersifat non volatile yang digunakan dalam suatu sistem mikroprosesor, yaitu :[3)]

1. Read Only Memory (ROM)
2. Programmable Read Only Memory (PROM)
3. Erasable Programmabel Read Only Memory (EPROM)
4. Electricity Alternate Read Only Memory (EAROM)

Untuk PROM, EPROM, dan EAROM data atau informasi dimasukkan ke dalam memory chip dengan suatu alat yang dapat memberikan pulsa tegangan yang cukup tinggi ke dalam sel-sel memory chip. Pada EPROM informasi yang telah disimpan masih dapat dihapus dengan sinar ultra ungu (*ultra violet*) dengan intensitas tertentu pada jendela (*window*) dari IC pada EPROM.

2.1.4.1.1 MEMBACA DATA DARI ROM

Urut-urutan proses pembacaan data dari ROM adalah sebagai berikut :[4)]

1. CPU memberikan input address kepada ROM sesuai dengan lokasi yang akan dibaca
2. CPU menunggu untuk selang waktu tertentu, berkisar antara 100-840 nano detik. Selang waktu ini diperlukan oleh rangkaian di dalam ROM untuk men-dekode address yang diterima dan data bus

---

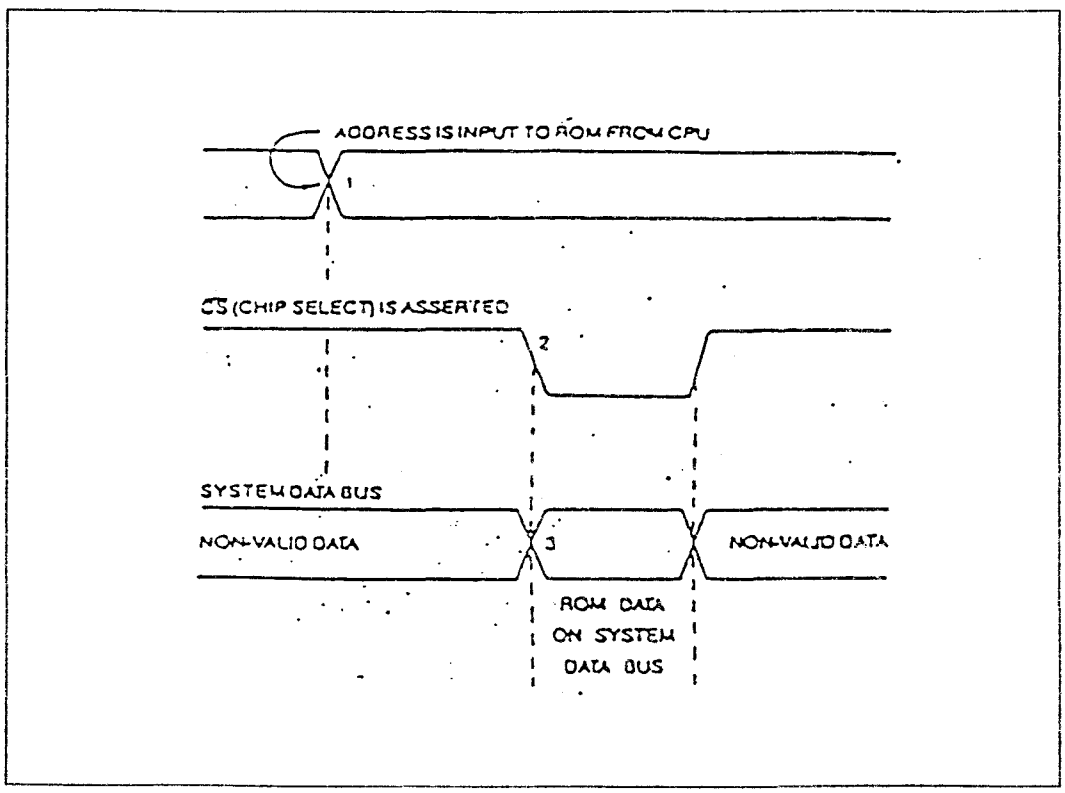
[3)] Coffron, James W., Z80 APPLICATION, Sybex Inc., 1983, hal. 1

[4)] Ibid, hal. 5

berada dalam keadaan siap menerima data

3. Sinyal Chip Select tidak diaktifkan untuk meniadakan data dari ROM pada data bus, sehingga alamat dan jalur pada ROM pada keadaan impedansi tinggi.

Gambar 2-3 menunjukkan diagram waktu untuk urutan pembacaan data dari ROM.



GAMBAR 2-3[5)]

DIAGRAM WAKTU PEMBACAAN DATA ROM

2.1.4.2 RANDOM ACCESS MEMORY

Random Access Memory (RAM) merupakan sistem

[5)] Ibid, hal. 6

memory yang menyimpan data hanya untuk sementara waktu (*volatile*), karena bila tegangan supply dimatikan maka data yang disimpan akan hilang. Dinamakan Random Access Memory karena lokasi manapun dapat dicapai secara langsung dengan menempatkan inputnya.

Terdapat 2 (dua) jenis RAM yang digunakan untuk menyimpan data, yaitu :

a. Static RAM

b. Dynamic RAM

Static RAM dapat menyimpan data dan tetap stabil selamanya selama daya tidak dimatikan, sedangkan Dynamic RAM harus diberi refresh cycle agar informasi dapat dipertahankan.

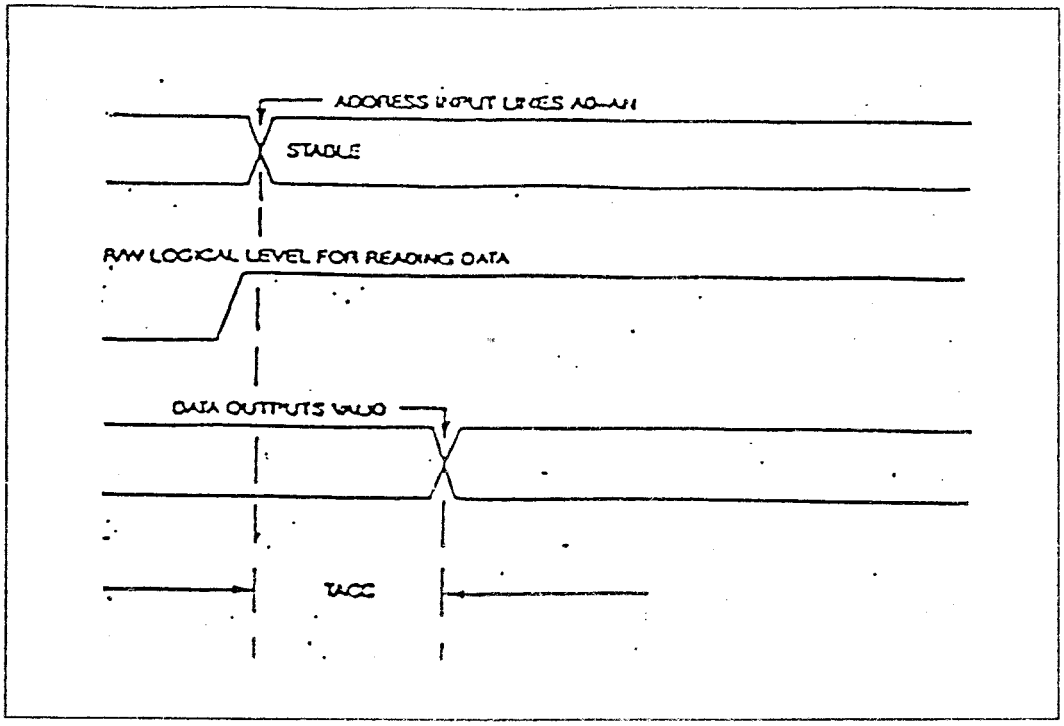
2.1.4.2.1 MEMBACA DATA DARI RAM

Urut-urutan sinyal yang diperlukan untuk membaca data dari RAM adalah sebagai berikut :[6]

1. Memory menerima address yang menentukan lokasi tertentu. Rangkaian dekoder dalam RAM memilih elemen manakah yang harus diaktifkan
2. Sinyal MEMR (Memory Read), menjadi aktif dan memory langsung menerima sinyal ini
3. Sistem menunggu dalam selang beberapa waktu tertentu (*Read Access Time*), sampai rangkaian di dalam RAM menjadi stabil

---

[6] Ibid, hal. 29



GAMBAR 2-4[7)]

DIAGRAM WAKTU PEMBACAAN DATA DARI RAM

4. Data akan dikirim dari memory ke data bus dan akan diterima oleh mikroprosesor. Jika mikroprosesor terlalu cepat mengaktifkan sinyal Chip Select (tanpa menunggu Read Access Time), maka mikroprosesor akan menerima data yang salah. Gambar 2-4 menunjukkan diagram pembacaan data dari RAM.

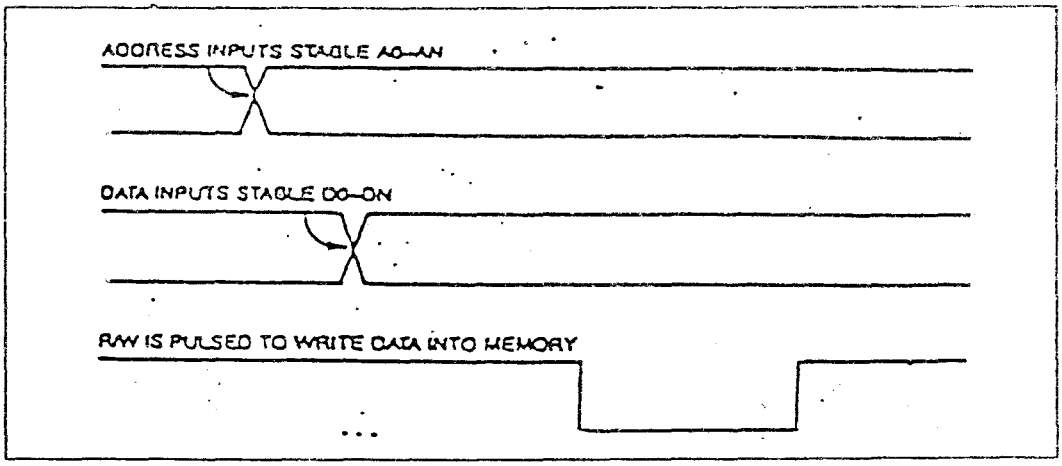
2.1.4.2.2 MENULIS DATA KE RAM

Diagram waktu penulisan data ke RAM, seperti

[7)] Ibid, hal. 29

ditunjukkan dalam gambar 2-5, proses penulisannya adalah sebagai berikut :[8)]

1. Address dari data yang akan ditulis dimasukkan ke dalam address memory yang sesuai dengan lokasi yang ditentukan
2. Data yang akan diisi ke memory diletakkan ke data bus
3. Mikroprosesor menunggu untuk selang waktu tertentu (*Write Access Time*) sampai rangkaian dalam RAM menjadi stabil
4. Setelah Write Access Time, sinyal memory write (MEMW) diaktifkan sehingga data yang terdapat pada input RAM tertulis ke dalam elemen penyimpan pada RAM.



GAMBAR 2-5[9)]

DIAGRAM WAKTU PENULISAN DATA KE RAM

[8)] Ibid, hal. 30

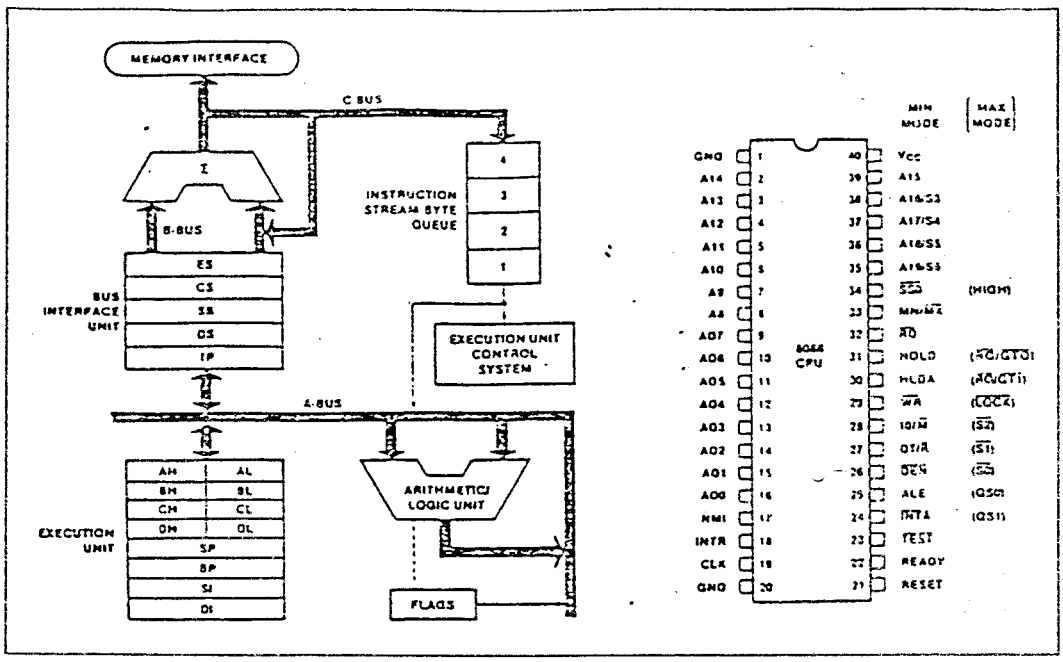
[9)] Ibid, hal. 30

2.2 MIKROPROSESOR 8088

Mikroprosesor 8088 merupakan turunan dari mikroprosesor 8086, hanya berbeda pada lebar jalur data busnya. Mikroprosesor 8086 mempunyai 16 bit internal data path maupun eksternal data bus, sedangkan mikroprosesor 8088 mempunyai internal data path 16 bit tetapi pada bagian eksternalnya hanya terdapat 8 bit data bus. Meskipun demikian software untuk mikroprosesor 8086 dan 8088 adalah sama hanya berbeda pada kecepatan eksekusi pada pengaksesan data 16 bit.

2.2.1 ARSITEKTUR INTERNAL MIKROPROSESOR 8088

Mikroprosesor 8088 mempunyai kemampuan pengalamatan 20 bit yang memungkinkan pengaksesan 1.048.576 byte memory. Akan tetapi karena instruksi-instruksinya hanya mengizinkan operasi dan manipulasi address 16 bit, maka seakan-akan hanya 65.536 byte saja yang dapat diakses. Pada gambar 2-6 dapat dilihat diagram blok dari rangkain internal 8088 dan juga pin-layoutnya. Mikroprosesor 8088 dapat dioperasikan dalam dua mode yaitu dalam mode minimum dan mode maksimum yang diatur oleh kondisi dari pin MN/MX. Apabila pin MN/$\overline{MX}$ berlogika high maka mode operasinya adalah minimum. Dan apabila pin MN/$\overline{MX}$ berlogika low, maka mikroprosesor 8088 dioperasikan dalam mode



GAMBAR 2-6[10)]

PIN OUT DAN ORGANISASI INTERNAL MIKROPROSESOR 8088

maksimum.

2.2.2 REGISTER 8088

Mikroprosesor 8088 mempunyai 14 buah register 16 bit. Register-register ini dikelompokkan atas 4 buah group, yaitu :

- Data register
- Pointer dan index register
- Segment register
- Instruction Pointer dan Flag register

---

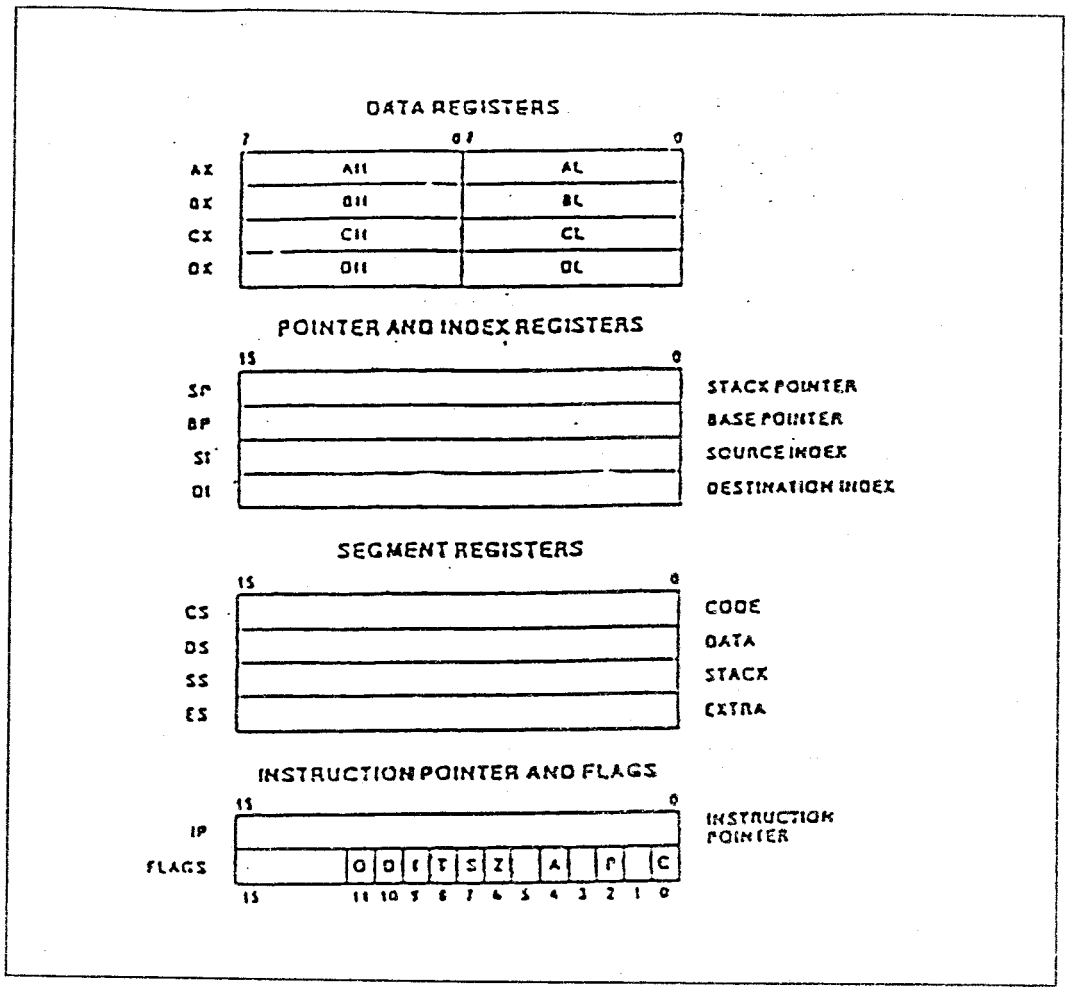
10) _____, MICROSYSTEM COMPONENTS HANDBOOK-MICROPROCESSOR VOLUME 1, Intel Co., 1985, hal. 3-105

Data register terdiri dari 4 buah register yaitu Accumulator register (AX), Base register (BX), Count register (CX) dan Data register (DX). Masing-masing register 16 bit ini dapat dioperasikan baik untuk operasi 16 bit data operasi (Word) atau dua-8 bit data operasi (byte). Jika register-register. dioperasikan untuk operasi 8 bit, maka setiap register akan terbagi menjadi dua register yaitu : AH, AL, BH, BL, CH, CL, DH, dan DL.

Register AX, BX, CX, dan DX memang disiapkan untuk register penggarap data, oleh karena itu register ini disebut sebagai register umum untuk menggarap (*General Purpose Register*). Instruction pointer register merupakan register 16 bit yang berfungsi mencatat offset address dari lokasi memory instruksi yang berikutnya pada harga base code segment saat itu.

Pointer dan index register terdiri dari 4 buah register 16 bit yaitu : SP (Stack Pointer), BP (Base Pointer), SI (Source Index), DI (Destination Index). Register-register ini hanya dapat dioperasikan untuk 16 bit. Fungsi dari register-register index dan pointer adalah pembentukan memory *Effective Address* (EA). Penggunaan register 8Ø88 ditunjukkan pada tabel 2-1.

Mikroprosesor 8Ø88 mencatat address untuk menunjukkan lokasi data di memory menggunakan segment



GAMBAR 2-7[11)]

REGISTER PADA MIKROPROSESOR 8088

register ditambah dengan offset address atau selisih address.

Kemampuan addressing dari 8088 adalah 20 bit yaitu 1.048.576 (1M) lokasi address, tetapi ada pula instruksi dari 8088 yang hanya dilaksanakan untuk 16 bit

---

11) Uffenbeck, J., THE 8086/8088 FAMILY : DESIGN, PROGRAMMING AND INTERFACING, Prentice-Hall Inc., 1987, hal. 33

TABEL 2-1 OPERASI REGISTER PADA MIKROPROSESOR 8088[12)]

REGISTER	O P E R A S I
AX	Perkalian word, pembagian word, I/O
AL	Perkalian byte, Pembagian byte, Translasi, Aritmatik desimal
AH	Perkalian byte, Pembagian byte
BX	Translasi
CX	Operasi String
DX	Perkalian word, Pembagian word, I/O
SP	Operasi stack
SI	Operasi string
DI	Operasi string

address. Cara yang dilakukan untuk mendapatkan 20 bit address dari 16 bit address adalah dengan menambahkan 16 bit offset address ke 16 bit segment address yang digeser ke kiri sebanyak 4 kali atau dengan kata lain segment addressnya dikalikan dengan 16 dan baru ditambahkan dengan offset address.

Flag register merupakan register pembantu terhadap semua operasi aritmatik dan logik. Bentuk bantuannya berupa menyimpan tanda keadaan operasi atau akibat operasi aritmatik yang terjadi. Jumlah bit pencata akibat operasi itu berjumlah 9 bit dan letak bit tersebut dan letak bit tersebut pada flag register sudah

---

[12)] Ibid, hal. 36

ditentukan secara pasti. Susunan bit dari flag register dapat dilihat pada gambar 2-8. Berikut penjelasan dari fungsi masing-masing bit.

- **CF (Carry Flag)**

Bit akan diset apabila terjadi carry out atau borrow pada high-order bit sebagai hasil operasi aritmatik.

- **OF (Overflow Flag)**

Bit ini akan diset apabila terjadi overflow atau hasil operasi aritmatika terlalu besar untuk daerah yang dituju.

- **SF (Sign Flag)**

Menunjukkan tanda dari bilangan hasil operasi aritmatik. Bila diset maka tanda bilangan adalah negatif.

- **PF (Parity Flag)**

Flag ini akan diset apabila hasil operasi mempunyai jumlah bit yang berlogika satu atau genap.

- **ZF (Zero Flag)**

Flag diset apabila hasil operasi aritmatik nol.

- **AF (Auxiliary Flag)**

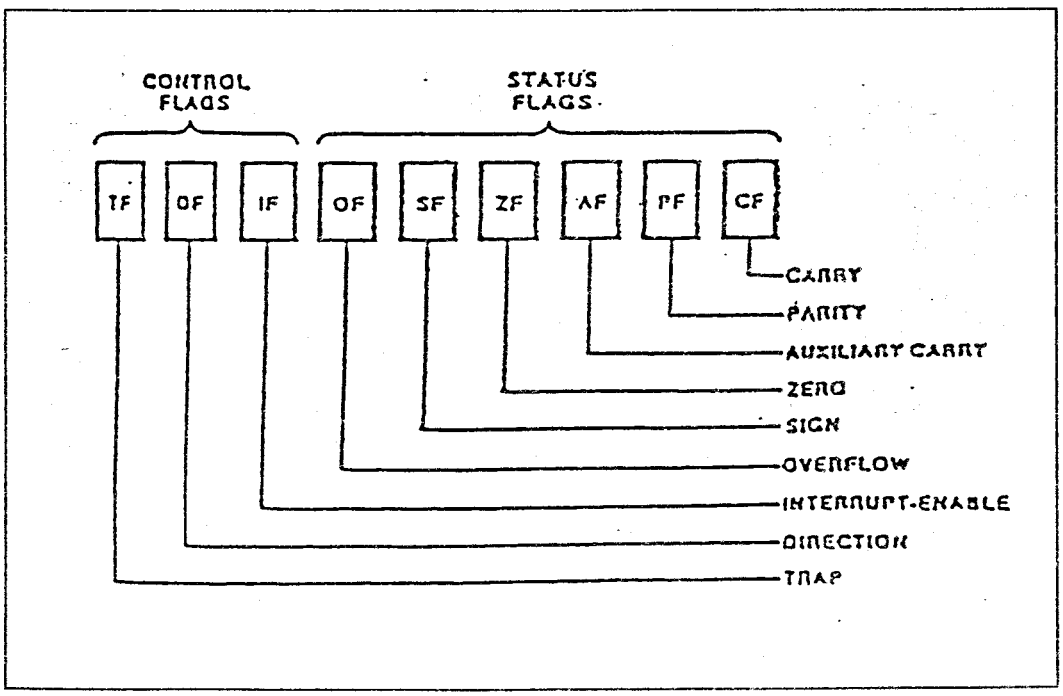
Bit ini akan diset apabila dalam operasi aritmatik terdapat carry out, borrow out dari nibble perhitungan desimal.

- DF (Direction Flag)

Jika bit ini diset maka operasi string akan melakukan proses pengurangan secara automatis dari high address ke low address. Jika bit ini dibuat nol, maka pada operasi string akan dilakukan penambahan secara automatis.

- IF (Interrupt Flag)

Apabila control flag diset, maka memungkinkan 8088 menerima external maskable interrupt.



GAMBAR 2-8[13)]

FLAG REGISTER MIKROPROSESOR 8088

- TF (Trap Flag)

Jika bit ini diset maka 8088 dalam mode single

[13)] Ibid, hal. 37

set. Setiap akhir dari sebuah instruksi, maka interrupt akan dibangkitkan secara automatis.

2.3 MIKROPROSESSOR 80286

Mikroprosesor Intel 8Ø286 adalah mikroprosesor 16 bit, dilengkapi dengan 14 register 16 bit, jalur data 16 bit dan jalur address 24 bit sehingga dapat menggarap memory hingga 1 pangkat 24 byte atau 16 Megabyte. Kecepatannya berkisar antara 1,4 hingga 2,1 MIP (Million Instruction per Second).

Mikroprosesor 8Ø286 memerlukan clock generator tipe 82284 dan bus controler tipe 82286 dan sebagai penunjang pengatur interupsi dapat menggunakan Programmable Interupt Controller tipe 8259.

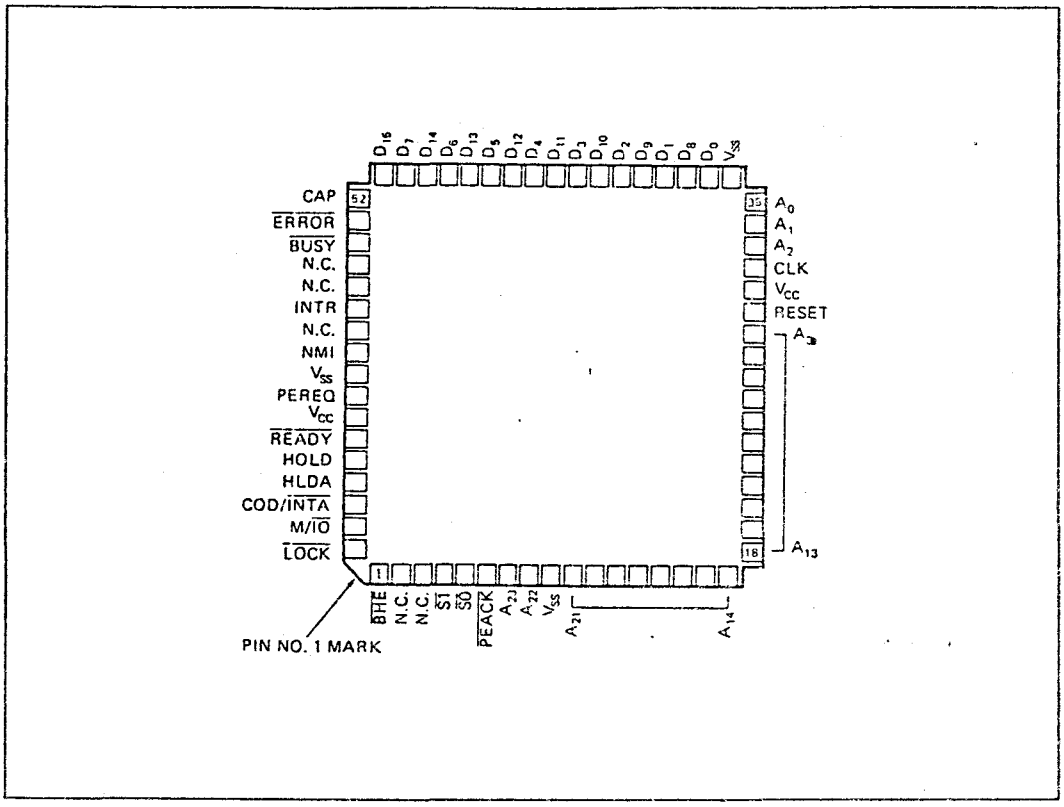
Bentuk fisik chip mikroprosessor 8Ø286 ada 3 macam, yaitu *Leadless Chip Carrier*, *Plastic Leaded Chip Carrier* dan *Pin Grid Array*. Jumlah kakinya 68 dan yang banyak digunakan di mikrokomputer adalah bentuk Pin Grid Array dengan susunan kaki seperti tampak pada gambar 2-9.

Ukuran sisi-sisinya 25,591 mm x 25,591 mm dan tebalnya 3,Ø99 mm. Jarak antara tiap kaki (tiap pin) tetap seperti jarak antar kaki IC, yaitu pada sisi tempat kaki chip dipasang.

Secara perangkat keras, 8Ø286 dapat dibagi

menjadi 4 bagian :

- **Execution Unit (EU)** terdiri dari semua register, ALU dan kontrolnya



GAMBAR 2-9[14)]

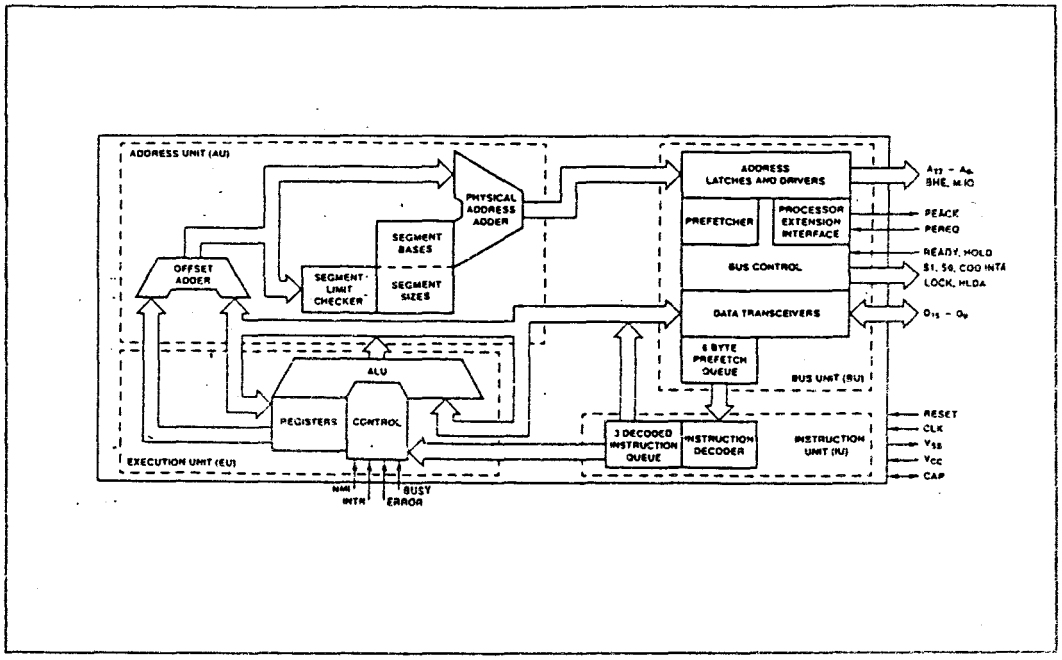
DIAGRAM PIN MIKROPROSESOR 80286

- **Bus Unit (BU)** terdiri dari Bus Control, dan perangkat prefetch queue
- **Instruction Unit (IU)** terdiri dari Instruction Decoder untuk queue
- **Address Unit (AU)** terdiri dari perangkat penghitung Address

---

14) Hall, Douglas V, Microprocessor and Interfacing, Mc Grav-Hill Inc, 1986, hal. 503

Agar lebih jelasnya tentang unit-unit tersebut di atas, dapat dilihat pada gambar 2-10.



GAMBAR 2-10[15)]

DIAGRAM BLOK INTERNAL MIKROPROSESOR 80286

Secara garis besar, susunan 8Ø286 dapat digambarkan dengan model sebagai berikut:

Instruction Unit (IU) adalah pengarahan *pipe lining* byte kode bahasa mesin yang dipungut oleh Bus Interface Unit (BIU). Pungutannya berjumlah 6 byte beruntun dan di-dekode-kan secara paralel di Instruction Unit.

Cara tersebut mempercepat eksekusi dibandingkan dengan mikroprosesor 8Ø86 atau 8Ø88 yang men-dekode-kan

---

15) Ibid, hal 502

byte instruksi di Execution Unit. Bus Unit (BU) dan Address Unit (AU) bekerja dengan bentuk operasi yang sama dengan Bus Interface Unit (BIU) pada mikroprosesor 8086 atau 8088.

Execution Unit (EU) mikroprosesor 80286 melaksanakan eksekusi yang berupa superset eksekusi pada mikroprosesor 8086 atau 8088. Artinya EU tidak hanya melaksanakan semua yang ada pada mikroprosesor 8086 atau 8088, akan tetapi juga memiliki kelebihan.

Secara umum 80286 dapat bekerja pada dua mode, yaitu :

- **Mode Riil (Real Address Mode)**
- **Mode address Viruil dengan Proteksi (Protected Virtual Address Mode).**

Kedua mode tersebut semuanya berupa superset pelaksanaan semua instruksi mikroprosesor 8086 atau 8088 pada taraf bahasa mesin dan bahasa assembly. Superset pelaksanaan berarti tidak hanya melaksanakan semua instruksi, akan tetapi juga dilengkapi dengan instruksi lain yang melengkapi operasinya.

Operasi 80286 pada mode adres riil (*real address mode*), menggarap lintas data memori secara riil. Yaitu secara fisik menggunakan 20 bit address melalui saluran A_0 - A_{19}, didampingi saluran BHE (*Bus High Enable*).

Operasi real mode menjangkau memori 1 Mega byte

(1Ø48576 byte) dengan perilaku tepata seperti mikroprosesor 8Ø86, menggunakan adres ber-segment dilengkapi adres ofset. Secara perangkat fisik dan perangkat lunak dapat dikatakan melakukan emulasi mikroprosesor 8Ø86 sepenuhnya.

Dengan demikian maka secara perangkat lunak mikroprosesor 8Ø286 juga melekukan emulasi mikroprosesor 8Ø88, sehingga taraf kompatibilitas pada aras perangkat lunak secara sepenuhnya kompatibel dengan mikroprosesor 8Ø86 atau 8Ø88 hingga taraf bahasa mesinnya. Dan operasi mikroprosesor 8Ø286 melaksanakan operasi jauh lebih cepat dengan ditambah atau dilengkapi dengan instruksi yang lebih canggih.

Sistem pelayanan operasi PC-DOS atau MS-DOS melakukan operasi pada mode adres riil (real mode). Pada operasi ini dapat diterapkan program yang melaksanakan lintas data aras dasar (*low level I/O*) yang searas dengan BIOS.

Misalnya, menggarap port-port periferal, lintas data secara langsung, dan lintas data komunikasi perangkat. Mode operasi ini hanya diperuntukkan bagi tugas tunggal (*single tasking*).

Salah satu yang membedakan 8Ø286 dengan 8Ø86 secara operasional adalah bahwa 8Ø286 dilengkapi dengan *Machine Status Word Register*. Jika bit Protection Enable

(bit PE) pada register ini diberi logika 1 (diset) menggunakan instruksi LMSW (*Load Machine Status Word*), maka 80286 beroperasi pada protected virtual address mode.

Operasi 80286 pada mode virtuil dengan proteksi (*Protected Virtual Address Mode*), menggarap adres memori secara virtuil, lengkap dengan operasi mode proteksi memori.

Yaitu secara fisik sepenuhnya menggunakan semua 24 bit adres yang ada melalui saluran A_0 - A_{23}, didampingi saluran BHE (*Bus High Enable*). Dalam keadaan operasi ini jumlah memori riil yang dapat dijangkau adalah 2 pangkat 24 byte atau 16 Mega byte (16 x 1024 x 1024 byte).

Selanjutnya penggarapan adres memori masih dapat dilaksanakan secara virtuil menggunakan pointer 32 bit yang dirinci menjadi pointer selector 16 bit dan pointer ofset 16 bit.

Proteksi memori diperlukan sekali jika komputer harus bekerja dengan pelaksanaan ganda (*multitasking*). Pelaksanaan ganda berarti sejumlah program pelaksanaan yang berbeda harus berada (berkoeksistensi) di memori pada lokasi yang berbeda.

Bagi setiap alokasi diperlukan proteksi, agar program yang satu tidak mengganggu alokasi program yang

lain. Sistem pelayanan UNIX dan OS/2 adalah sistem pelayanan operasi yang melayani pelayanan ganda (*multitasking*) yang memerlukan mode proteksi tersebut.

2.3.1 MEMORY DAN I/O MAPPING

TABEL 2-2 I/O Map Komputer IBM PC AT[16]

Hex Range	Device
000 - 01F	DMA Controller 1, 8237A-5
020 - 03F	Interrupt Controller 1, 8259A, Master
040 - 05F	Timer, 8254, 2
060 - 06F	8042 (Keyboard)
070 - 07F	Real Time Clock, NMI
080 - 08F	DMA Page Register, 74LS612
0A0 - 0BF	Interrupt Controller 2, 8259A
0C0 - 0DF	DMA Controller 2, 8237-5
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8	Math Coprocessor
1F0 - 1F8	Fixed Disk
200 - 207	Game I/O
278 - 27F	Parallel Printer Port 2
2F8 - 2FF	Serial Port 2
300 - 31F	Prototype Card
360 - 36F	Reserved
378 - 37F	Parallel Printer Port 1
380 - 38F	SDLC, Bisynchronous 2
3A0 - 3AF	Bisynchronous 1
3B0 - 3BF	Monochrome Display and Printer Adapter
3C0 - 3CF	Reserved
3D0 - 3DF	Color/Graphics Monitor Adapter
3F0 - 3F7	Diskette Controller
3F8 - 3FF	Serial Port

---

[16] ....., IBM PC AT Technical Reference, hal. I-28

Sistem memory map dari komputer IBM PC AT ditunjukkan seperti pada tabel 2-2. Sedangkan I/O map dari komputer IBM PC AT ditunjukkan pada tabel berikut:

TABEL 2-3 MEMORY MAP KOMPUTER IBM PC AT[17)]

Address	Name	Function
000000 to 07FFFF	512 KB System Board	System Board Memory
080000 to 09FFFF	128 KB	I/O Channel Memory - IBM Personal Computer AT 128 KB Memory Expansion Op - tion
0A0000 to 0BFFFF	128 KB Video RAM	Reserved for Graphics Display Buffer
0C0000 to 0DFFFF	128 KB I/O Expansion ROM	Reserved for ROM on I/O Adapter
0E0000 to 0EFFFF	64 KB Reserved on System Board	Duplicated Code Assign - ment at Address FE0000
0F0000 to 0FFFFF	64 KB ROM on the System Board	Duplicated Code Assign - ment at Address FF0000
100000 to FDFFFF	Maximum Memory 15 MB	I/O Channel Memory - IBM Personal Computer AT 512 KB Memory Expansion Option
FE0000 t0 FEFFFF	64 KB Reserved on System	Duplicated Code Assign - ment at Address 0E0000
FF0000 to FFFFFF	64 KB ROM on the System Board	Duplicated Code Assign - ment at Address 0F0000

---

17) Ibid, hal. 1-8

2.4 TRANSMISI DATA ANTAR KOMPUTER

Transmisi adalah suatu perpindahan informasi dari suatu tempat ke tempat lainnya. Suatu sistem transmisi yang lengkap akan mengandung suatu pemancar atau transmitter, medium pentransmisi dimana informasi ditransmisikan, dan receiver yang akan menghasilkan salinan output informasi pada tempat tujuan. Pada waktu perpindahan informasi melewati medium transmisi akan mengalami gangguan seperti bising atau noise serta sinyal-sinyal interferensi lainnya.

Dalam transmisi data dikenal tiga istilah yaitu *Simplex, Half-Duplex dan Full-Duplex*[18]. Pada transmisi data Simplex, data hanya dikirim dalam satu arah, sedangkan pada transmisi data Half-Dufplex data dapat ditransmisikan dalam dua arah tetapi secara bergantian. Sedangkan transmisi data Full-Duplex merupakan transmisi data dua arah dimana data diterima sistem sekaligus mengirimkan data dalam waktu yang sama.

Berdasarkan bentuk sinyal yang dikirim, transmisi dibagi menjadi dua, yaitu transmisi analog dan transmisi digital. Transmisi analog adalah transmisi sinyal secara kontinyu, seperti sinyal bunyi atau suara. Transmisi analog sangat peka terhadap noise dan distorsi. Transmisi digital adalah transmisi sinyal yang berupa

---

[18] Hall, D.V., Microprocessors and Interfacing: Programming and Hardware, McGraw-Hill Book Company, Singapore, 1987, hal. 442

aliran pulsa ON dan OFF atau tegangan listrik dengan kondisi logika '0' dan '1'. Pulsa tersebut dikenal dengan sebutan binary digit (bit).

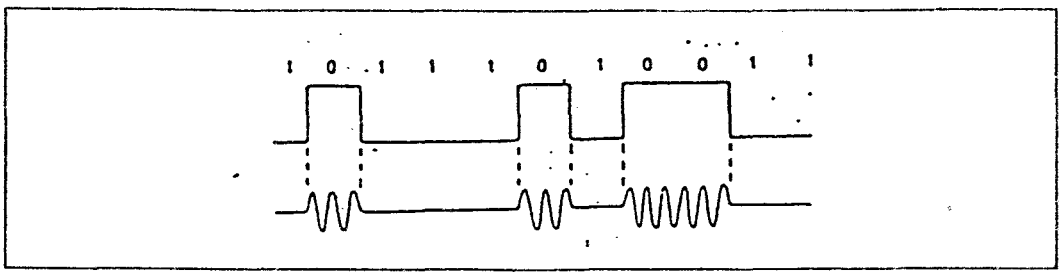
Sistem transmisi data dapat menggunakan transmisi analog maupun transmisi digital. Pada komunikasi data melalui kabel telephone atau udara, sinyal digital pada komputer harus diubah menjadi sinyal analog terlebih dahulu.

2.4.1 METODE MODULASI

Pada dasarnya ada tiga metoda untuk memodulasi data ke gelombang pembawa. Cara yang pertama adalah modulasi amplitudo (Amplitudo Modulation, AM), amplitudo gelombang pembawa divariasi relatif terhadap pola bit yang ditransmisi. Bentuk yang paling sederhana adalah ON-OFF keying. Di sini logika '1' ditransmisikan sebagai level rendah (OFF) dan logika '0' sebagai level tinggi (ON). Seperti yang terlihat pada gambar 2-11 dan gambar 2-12.

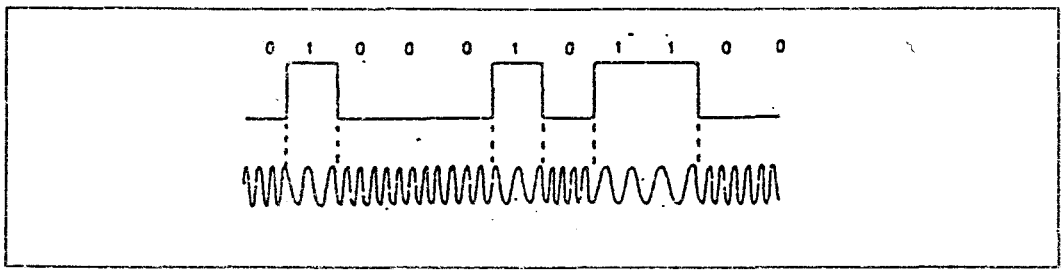
Metode kedua, berupa bentuk sederhana dari modulasi frekwensi (FM) yang disebut *Frekwensi Shift Keying* (FSK). Pengiriman logika '1' atau logika '0' dilakukan denga berpindah-pindah antara frekwensi yang lebih rendah ke frekwensi yang lebih tinggi. Tentunya

kedua frekwensi tersebut harus berada di dalam frekwensi jalur.



GAMBAR 2-11[19]

AM/ON-OFF KEYING



GAMBAR 2-12[20]

FM/FREKWENSI SHIFT KEYING

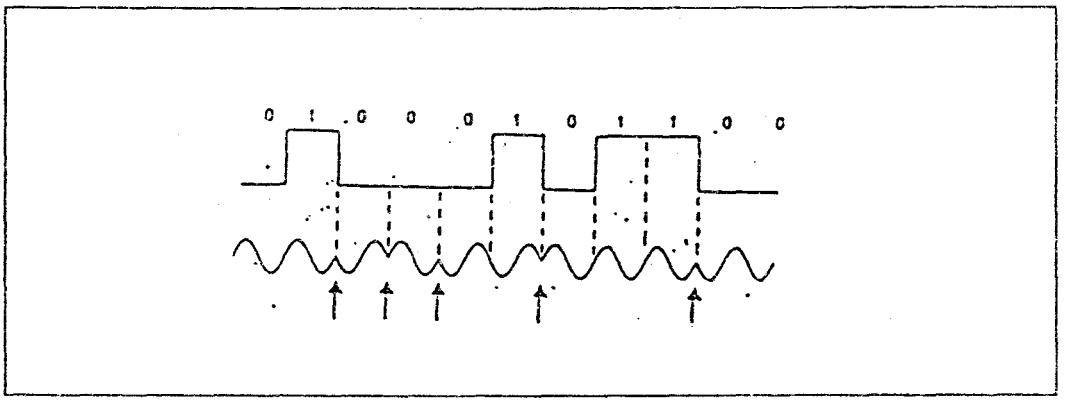
Metode ketiga adalah modulasi fase (PM). Metode ini sangat baik untuk mengirim data. Untuk mengirim sebuah logika '0', gelombang pengirim data menghasilkan fase 180^{0} misalnya, dan bila mengirim logika '1', fase yang dihasilkan tidak berubah. Seperti yang terlihat

---

19) Den Heijer, P.C., Komunikasi Data, Elex Media Komputindo, 1988, hal. 61

20) Den Heijer, P.C., Loc.cit.

pada gambar 2-13. Metoda ini terutama sesuai untuk modem-modem yang kecepatan sinyalnya harus lebih tinggi dari tingkat Baud saluran transmisi. Sehingga kemungkinan akan membutuhkan empat sampai delapan perubahan fase yang berbeda. Dalam modulasi 4-fase, perubahan fase dilaksanakan setiap dua bit. Oleh karena itu, setiap perubahan gelombang pembawa (BAUD), yang dikirimkan adalah jumlah kelipatan bit (bps). Dengan demikian, jalur dengan tingat Baud yang dibatasi oleh lebar band, dapat digunakan untuk kecepatan sinyal yang lebih tinggi.



GAMBAR 2-13[21)]

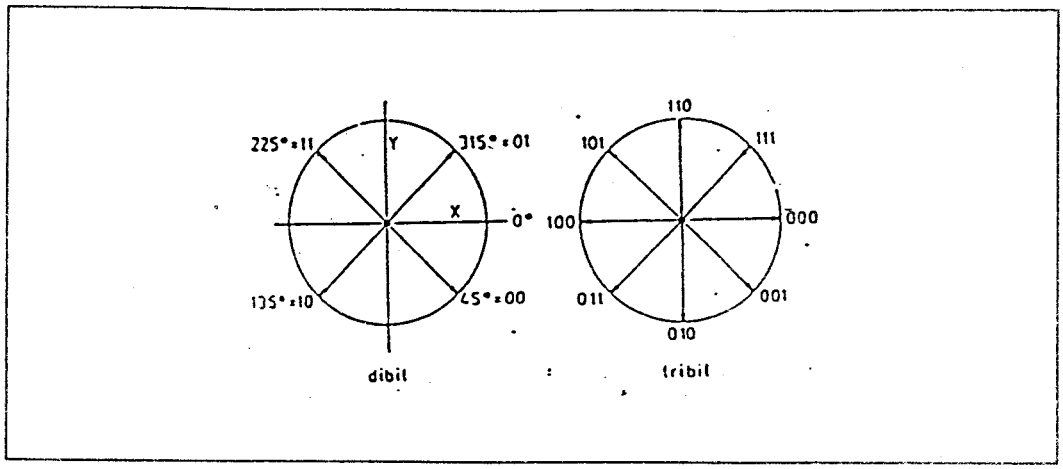
MODULASI FASE

Di samping modulasi DIBIT (transmisi dua bit) ini, dapat pula digunakan modulasi TRIBIT. Dalam hal ini dipilih salah satu di antara kedelapan perubahan fase

---

[21)] Ibid, hal. 62

yang mungkin untuk mengirim tiga bit data[22], seperti yang diperlihatkan pada gambar 2-14.



GAMBAR 2-14[23]

MODULASI FASE DIBIT DAN TRIBIT

Dalam suatu sistem mikrokomputer, transmisi data selalu dilaksanakan secara paralel. Karena hal tersebut merupakan cara yang tercepat yang masih dapat dilakukan. Namun untuk transmisi jarak jauh, komunikasi data secara paralel akan membutuhkan banyak kabel, sehingga akan menimbulkan pemborosan. Oleh karena itu, transmisi data dengan jarak jauh data yang akan dikirimkan diubah dari bentuk paralel menjadi serial, sehingga data tersebut dapat dikirimkan dengan hanya melalui sepasang kabel. Data serial yang diterima kemudian diubah kembali ke dalam bentuk paralel sehingga data tersebut dengan mudah

---

[22] Ibid, hal. 62

[23] Ibid, hal. 62

dilewatkan pada bus komputer.

2.5 KOMUNIKASI ASYNCHRONOUS

Pada Tugas Akhir ini secara khusus dibahas mengenai transmisi data antar komputer secara serial. Adapun pada transmisi serial ini, data-data dikirimkan satu bit demi satu bit dalam bentuk kode biner, sebagai contoh untuk data-data yang setiap karakternya terdiri dari 8 bit maka 8 bit data ini akan dikirimkan satu per satu. Demikian pula pada sisi penerima, karakter juga diterima 8 bit data satu per satu.

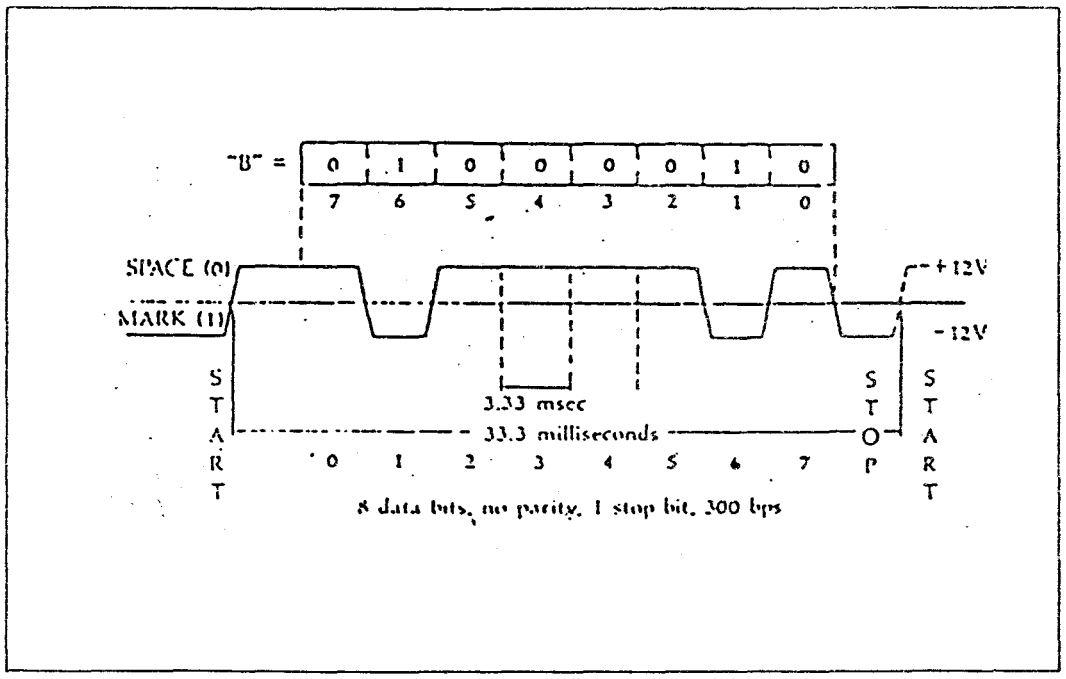
Pada pengiriman data secara serial, harus dilakukan sinkronisasi atau penyesuaian antara pengirim dengan penerima agar data yang dikirimkan dapat ditafsirkan secara tepat dan benar oleh penerima. Jadi dapat dikatakan fungsi sinkronisasi adalah :

- Agar penerima dapat mengetahui dengan tepat bilamana sinyal yang diterima merupakan bit dari suatu data (*sinkronisasi bit*).
- Agar penerima dapat mengetahui dengan tepat bit data yang membentuk sebuah karakter (*sinkronisasi karakter*)

Pada transmisi asinkron dilakukan pengiriman data berupa karakter yang dikirimkan satu per satu setiap kali pengirimannya. Antara satu karakter dengan karakter yang lainnya tidak terdapat waktu antara yang tetap. Karakter

dapat dikirimkan sekaligus ataupun beberapa karakter, kemudian berhenti untuk jangka waktu yang tidak tentu, kemudian mengirimkan sisanya. hal ini berakibat setiap kali penerima harus melakukan sinkronisasi agar bit data yang dikirimkan dapat diterima dengan benar, sehingga penerima harus mengetahui bit pertama dari sinyal data. Hal ini dilakukan dengan cara memberikan suatu pulsa yang disebut start pulse yaitu berupa bit awal (*start bit*) pada awal setiap karakter. Pulsa ini memberitahukan penerima untuk mulai menerima bit data. Secara singkat hal ini dapat dijelaskan sebagai berikut : satu frame karakter yang dikirimkan dimulai dengan bit awal (*start bit*) yang berkondisi logika "Ø" (*low*), memberitahukan sistem untuk mulai mengumpulkan bit-bit berikutnya sebagai bit-bit data, kemudian setelah bit-bit data tersebut lengkap diterima, bit akhir (*stop bit*) sebanyak 1, $1^1/_2$, atau 2 bit yang berkondisi logika '1' berfungsi untuk memberitahukan kepada terminal bahwa data telah lengkap dan terminal kembali ke keadaan reset untuk dapat menerima bit awal lagi. Keadaan kondisi reset ini disebut keadaan idle (*idle state*) yang berkondisi logika '1'. Untuk mencegah terdapatnya kesalahan pada data yang diterima, maka pada data yang dikirimkan ditambahkan bit paritas (*parity bit*) pada bagian akhir rangkaian bit-bit data, sebelum bit akhir (*stop bit*).

Pada komunikasi asynchronous harus dioperasikan pada kecepatan yang sama antara kedua sisi hubungan. Kecepatan transmisi data diukur dalam satuan *bit per second* (bps). Pada gambar 2-15 menunjukkan sinyal 300 bps, waktu yang dibutuhkan untuk mengirimkan setiap bit adalah 3,33 milidetik (1/300 detik). Istilah baud juga digunakan selain bps, tetapi dalam standard industri kedua pengertian tersebut mempunyai pengertian yang sama.[24)]

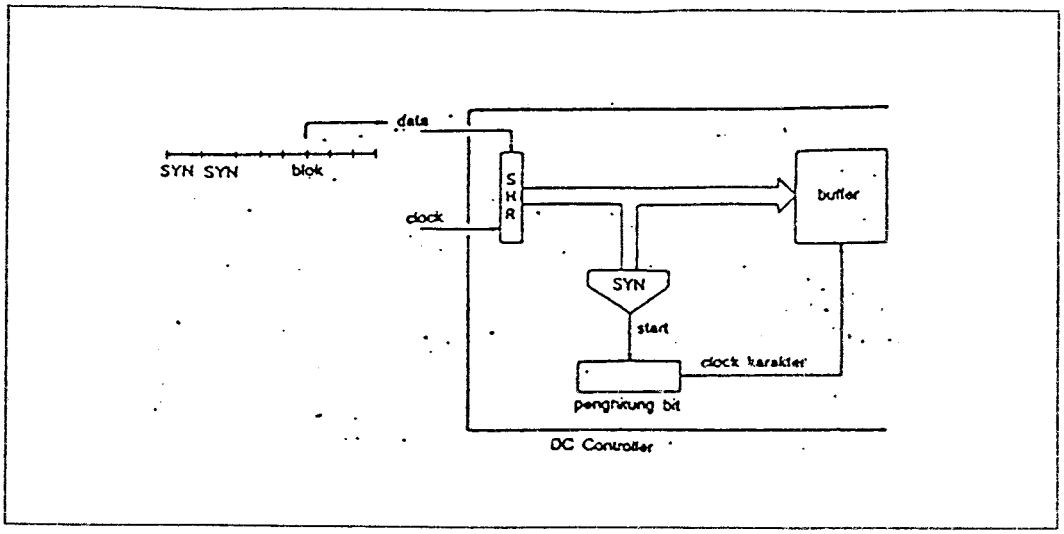


GAMBAR 2-15[25)]

FORMAT DATA KARAKTER TRANSMISI SERIAL ASINKRON

[24)] Kruglinski, David, Guide to IBM PC Communications, The Osborne/McGraw-Hill, 1986, hal. 41

[25)] Ibid, hal. 39



GAMBAR 2-16[26]

SINKRONISASI KARAKTER

2.6. INTERFACE SERIAL RS-232-C

Semua sinyal listrik yang disalurkan melalui media transmisi akan bertambah lemah karena adanya hambatan, kapasitansi serta induktansi media. Hal ini berakibat sinyal informasi dapat terganggu oleh derau dan terdapat kesulitan untuk membedakan keduanya. Penyaluran informasi jarak jauh membutuhkan penguatan atau cara lain agar informasi ini dapat tiba di tempat tujuan. Guna mencapai tujuan ini digunakan gelombang pembawa yang memodulasi sinyal informasi. Peralatan yang memungkinkan penyaluran informasi dengan gelombang pembawa dikenal sebagai modem. Modem dan peralatan lain yang digunakan untuk mengirimkan data serial disebut

---

[26] Heijer, Den, Op. Cit, hal. 28

sebagai Data Communication Equipment (DCE), dan terminal atau komputer yang mengirimkan atau menerima data disebut sebagai Data Terminal Equipment (DTE). Di samping itu, agar peralatan komunikasi data dapat saling berhubungan dibuatlah ketentuan baku. Electronic Industries Association (EIA) mengembangkan suatu standar yang disebut RS-232-C untuk pertukaran secara serial dengan sinyal biner antar DTE dan DCE. Negara-negara di luar Amerika menggunakan ketentuan baku yang mirip dengan RS-232-C, yang dikenal sebagai CCITT V24. Standar ini menjelaskan fungsi-fungsi dari keduapuluhlima sinyal dan pin-pin untuk pertukaran data secara serial.

2.6.1 FUNGSI-FUNGSI RS-232-C DAN CCITT V. 24

Untuk mengetahui lebih mendalam tentang RS-232-C dan CCITT V.24, berikut akan dibahas secara singkat tentang fungsi-fungsi rangkaian interface ini :[27)]

A. Protective Case Ground, pin 1

Pin ini harus dihubungkan secara listrik dengan frame peralatan, kemudian dapat disambungkan ke tanah (*external ground*), apabila diperlukan.

B. Transmitted Data (TD), pin 2

Sinyal dalam rangkaian ini dibangkitkan oleh DTE untuk diberikan kepada DCE untuk disalurkan melalui

---

27) Campbell, Joe, **THE RS-232 SOLUTION**, Second Edition, Berkeley, California, Campbell Productions, 1988, hal. 52

saluran komunikasi ke tujuannya. Ketentuan RS-232-C meminta agar DTE harus selalu dalam keadaan ON (MARK) pada saat tidak ada data yang dikirimkan atau pada interval antar karakter.

C. Received Data (RD), pin 3

Sinyal ini berisikan data yang dibangkitkan oleh DCE dalam sinyal yang diterima dari pengirim.

D. Request To Send (RTS), pin 4

Sinyal ini mengendalikan DCE pada bagian pengirim untuk menyalurkan data. Keadaan ON (MARK) menyebabkan rangkaian DCE untuk siap menerima data. Apabila DTE akan mengirimkan data, akan membuat rangkaian RTS ini ON. DCE dari pengirim harus siap mengirimkan data demikian pula DCE tujuan harus siap menerima data ini. Keadaan OFF menyatakan bahwa DCE tidak berada dalam keadaan siap untuk transmisi. Keadaan transisi dari OFF ke ON mengakibatkan DCE memasuki mode transmit (siap untuk mengirim). Untuk menyatakan bahwa DCE telah siap, rangkaian Clear To Send menjadi ON, sehingga DTE dapat mengirimkan datanya melalui rangkaian Transmitted Data. Transmisi data harus dilakukan sebelum RTS, CTS, DSR atau DTR menjadi OFF. Sebaliknya keadaan transisi dari ON ke OFF mengakibatkan DCE menyelesaikan transmisi data dan kembali keadaan non-transmit. DCE kemudian membuat

CTS menjadi OFF. DTE tidak boleh membuat rangkaian ini OFF sebelum bit terakhir disalurkan ke rangkaian Transmitted Data. Apabila RTS diset menjadi OFF maka RTS tidak boleh menjadi ON lagi sebelum CTS dibuat menjadi OFF oleh DCE. Keadaan ON atau OFF rangkaian CTS merupakan tanggapan atas keadaan ON atau OFF dari rangkaian RTS. Apabila rangkaian RTS atau CTS keduanya dalam keadaan OFF, maka DTE harus menjaga agar rangkaian Transmitted Data tetap mengirimkan sinyal berkondisi logika "1". Rangkaian RTS tidak boleh berubah menjadi ON lagi sampai rangkain CTS berubah menjadi OFF (oleh DCE).

E. **Clear To Send (CTS), pin 5**

Sinyal ini untuk menyatakan bahwa DCE siap untuk mengirimkan data melalui saluran komunikasi. Keadaan ON pada pin ini bersama dengan keadaan ON pada RTS, DSR, dan juga DTR menyatakan bahwa sinyal pada DTE akan ditransmisikan melalui saluran komunikasi. Keadaan OFF menyatakan pada DTE untuk tidak menyalurkan data melalui rangkaian Transmitted Data. Keadaan ON pada rangkaian ini merupakan jawaban atas terjadinya keadaan ON pada rangkaian DSR dan RTS secara bersamaan. Keadaan CTS ON tidak menjamin bahwa DCE penerima siap menerima data.

F. **Data Set Ready (DSR), pin 6**

Fungsi sinyal ini adalah untuk menyatakan status modem lokal yang tersambung pada DCE. Keadaan ON menunjukkan bahwa peralatan telah tersambung pada saluran dan DCE siap bertukar sinyal kendali dengan DTE untuk memulai pertukaran data.

G. **Signal Ground, pin 7**

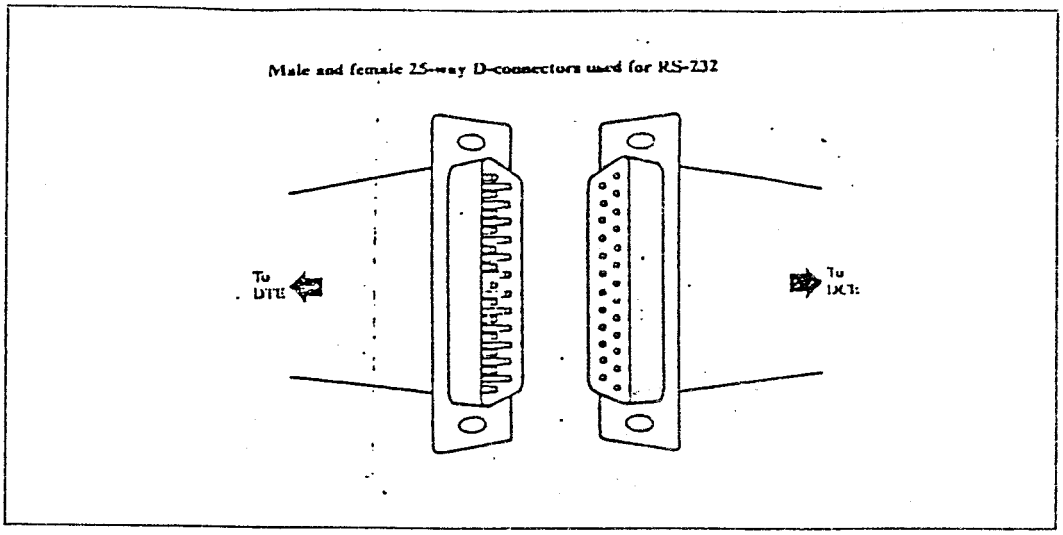
Pin ini merupakan tegangan acuan untuk semua rangkaian, kecuali Protective Ground. Dalam peralatan komunikasi data rangkaian ini harus disambungkan ke satu titik.

H. **Data Carrier Detect (DCD), pin 8**

Digunakan untuk menunjukkan bahwa peralatan terminal siap beroperasi. DCD ini kadang-kadang disebut juga Receive Line Signal Detector.

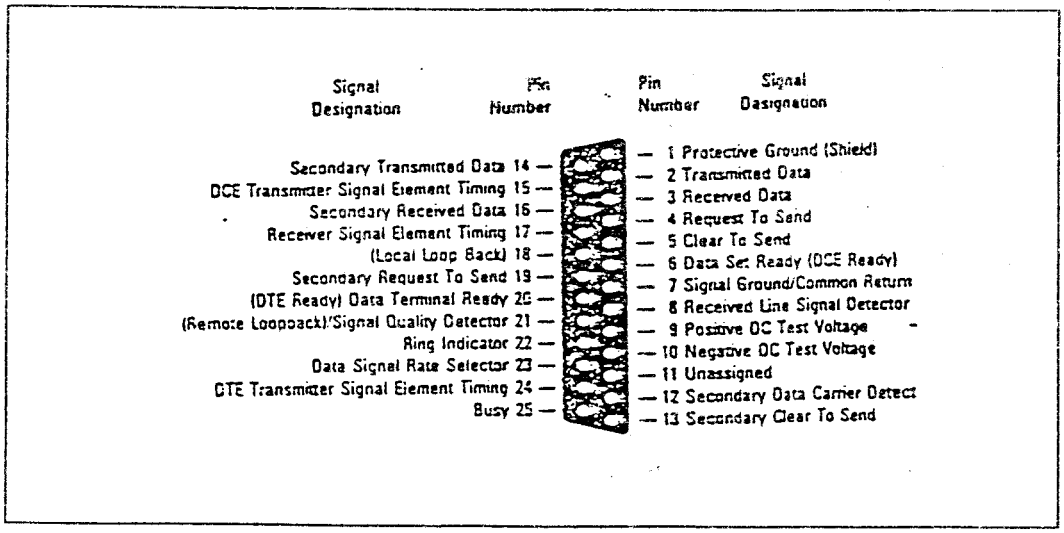
I. **Data Terminal Ready (DTR), pin 20**

Keadaan ON pada pin ini menunjukkan bahwa DTE siap untuk beroperasi dan menyiapkan DCE untuk menghubungkan diri dengan saluran komunikasi (misalnya dengan modem). DTE dapat memberikan sinyal ini apabila siap untuk melakukan transmisi atau menerima data. Keadaan OFF menyebabkan DCE dilepaskan dari saluran setelah data selesai disalurkan.



GAMBAR 2-17[28)]

KONEKTOR DB-25



GAMBAR 2-18[29)]

PIN-PIN PADA KONEKTOR DB-25

---

28) Tooley, Michael, DATA COMMUNICATION POCKET BOOK, Heinemann Ltd, 1989, hal. 72

29) Held, Gilbert, DATA COMMUNICATION NETWORKING DEVICE, Second Edition, Wiley Corp, 1988, hal. 68

J. RING INDIKATOR (RI), pin 22

Ring indikator diterima apabila modem menerima sinyal.

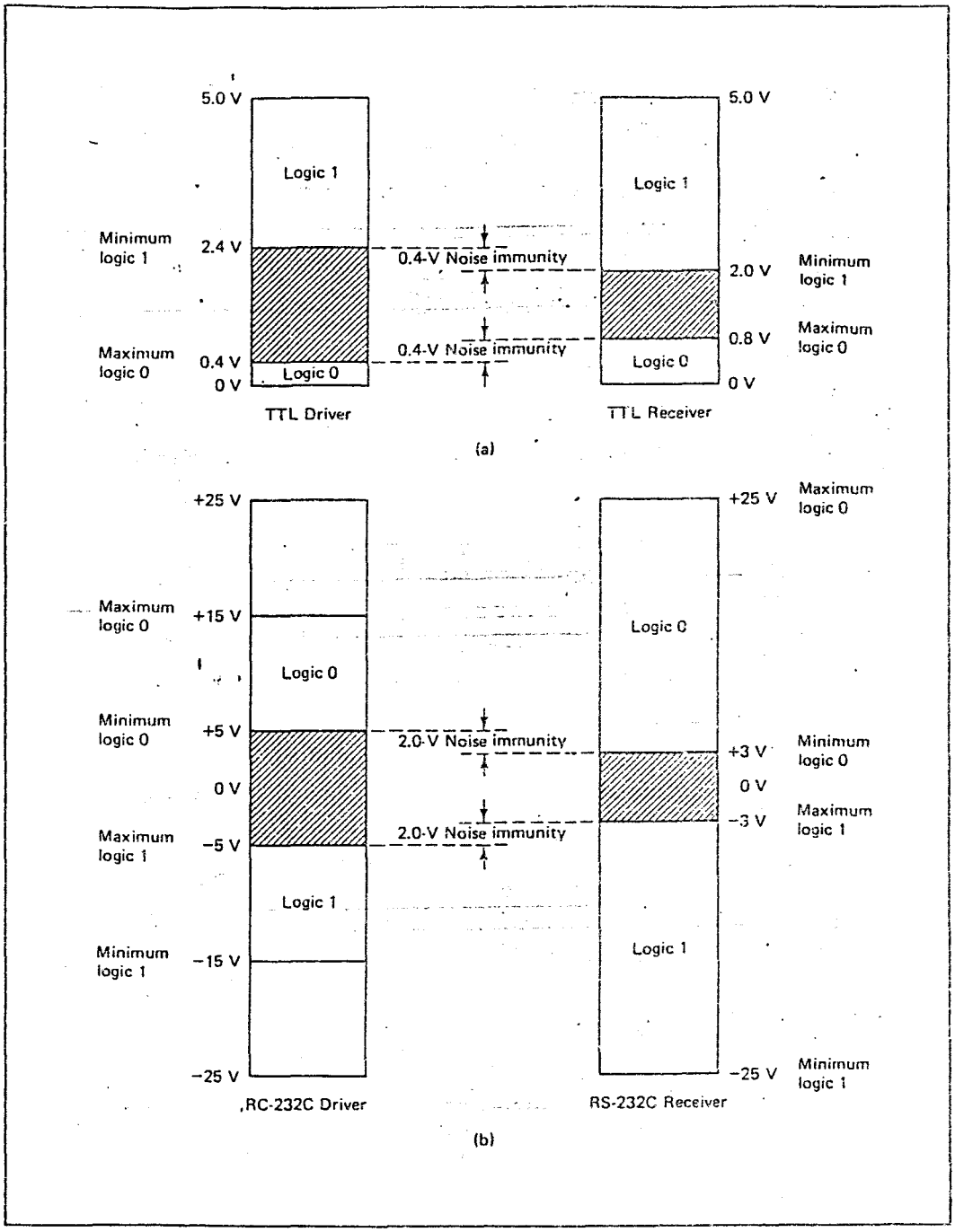
Pin-pin tersebut di atas selengkapnya merupakan pin-pin pada konektor DB-25 yang merupakan penghubung / konektor pada hubungan antar komputer melalui interface serial RS-232-C, seperti yang terlihat pada gambar 2-17 dan gambar 2-18.

2.6.2 Level Tegangan Sinyal Pada RS-232-C

Rangkaian Universal Asynchronous Receiver Transmitter (UART) berfungsi sebagai *asynchronous communication adapter*, yaitu antara lain sebagai pengubah data-data paralel menjadi serial. Pada tugas akhir ini digunakan IC INS 8250. Sinyal keluaran dari IC INS 8250 ini mempunyai level tegangan TTL; yaitu sinyal dengan level tegangan berkisar antara 0 volt (untuk kondisi logika '0') dan +5 volt (untuk kondisi logika '1'). Pada standar komunikasi serial sesuai dengan spesifikasi standar RS-232-C; level tegangan yang digunakan ialah level tegangan yang berkisar antara -3 volt hingga -15 volt untuk kondisi logika '1' atau yang disebut dengan keadaan "*mark*" dan antara +3 volt hingga +15 volt untuk kondisi logika '0' atau yang disebut dengan "*space*"; atau dengan kata lain standar RS-232-C

menggunakan logika negatif/ terbalik. Untuk menyesuaikan level tegangan keluaran USART (IC INS 8250) tersebut dengan level tegangan sesuai dengan standar RS-232-C diperlukan rangkaian pengubah tegangan (*voltage translator*), yang disebut dengan *line drivers* dan *line receivers*, yang berfungsi sebagai interface rangkaian dengan level tegangan TTL dengan sinyal RS-232-C. Pengubahan level tegangan ini dilakukan oleh IC MC-1488, sebagai line driver dan IC MC-1489 sebagai line receiver; di mana IC MC-1488 akan mengubah level tegangan keluaran dari IC 8250 sebesar 0 volt (kondisi logika '0') menjadi +15 volt (*space*) dan level tegangan 5 volt (kondisi logika '1') menjadi -15 volt (*mark*), yaitu level RS-232-C. Sebaliknya pengubahan level sinyal masukan RS-232-C ke level TTL dilakukan oleh IC MC-1489.

Salah satu keuntungan pemindahan kata lewat kabel dengan level RS-232C adalah kekebalan yang tinggi terhadap noise yang dapat timbul pada jalur transmisi. Gambar 2-21 menunjukkan pengaruh noise terhadap bentuk sinyal. Terlihat bahwa adanya noise tidak mempengaruhi keadaan logika dari sinyal. Setelah diubah menjadi level tegangan TTL maka sinyal dalam keadaan baik (tanpa noise).

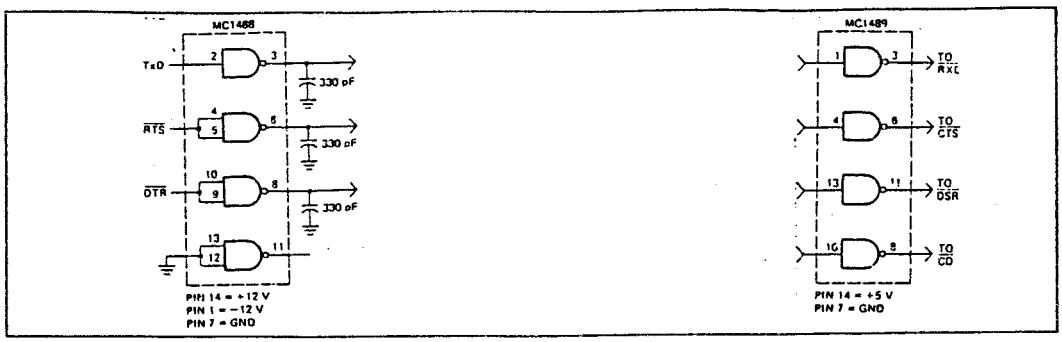


GAMBAR 2-19[30)]

PERBANDINGAN LEVEL TEGANGAN TTL DAN RS-232-C

---

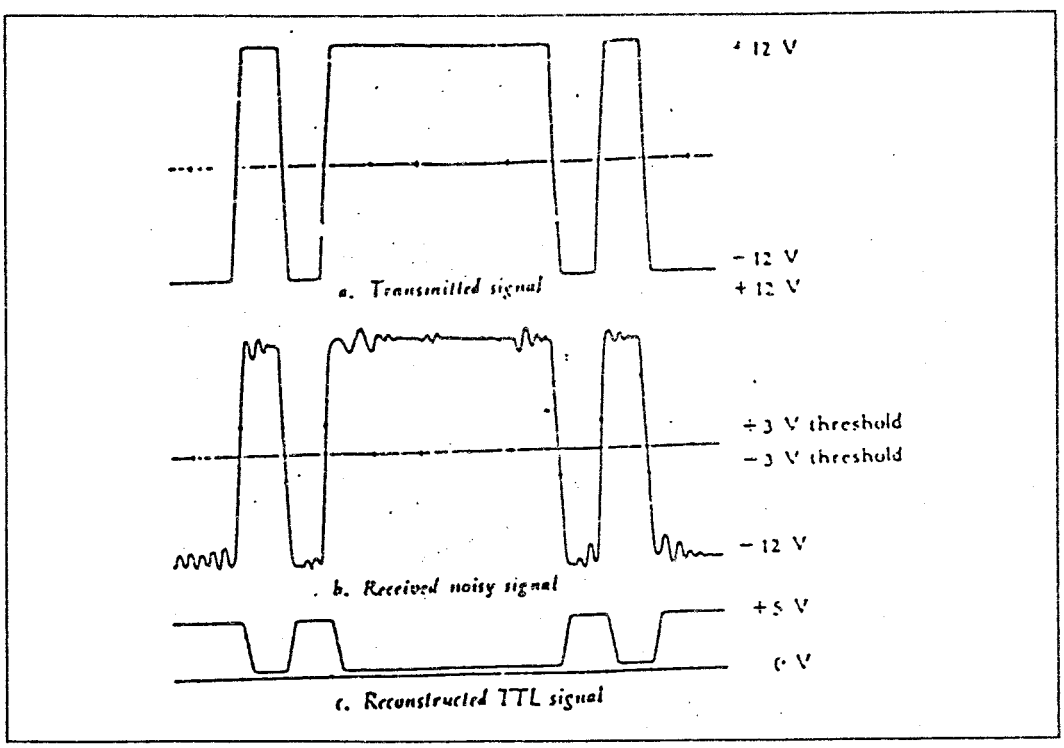
30) Uffenbeck, John, THE 8086/8088 FAMILY : DESIGN, PROGRAMMING AND INTERFACING, Prentice-Hall Inc., 1987, hal. 479



A) TTL - RS-232 B) RS-232 - TTL

GAMBAR 2-20[31)]

PERUBAHAN LEVEL RS - TTL DAN TTL - RS



GAMBAR 2-21[32)]

PENGARUH NOISE PADA SINYAL

---

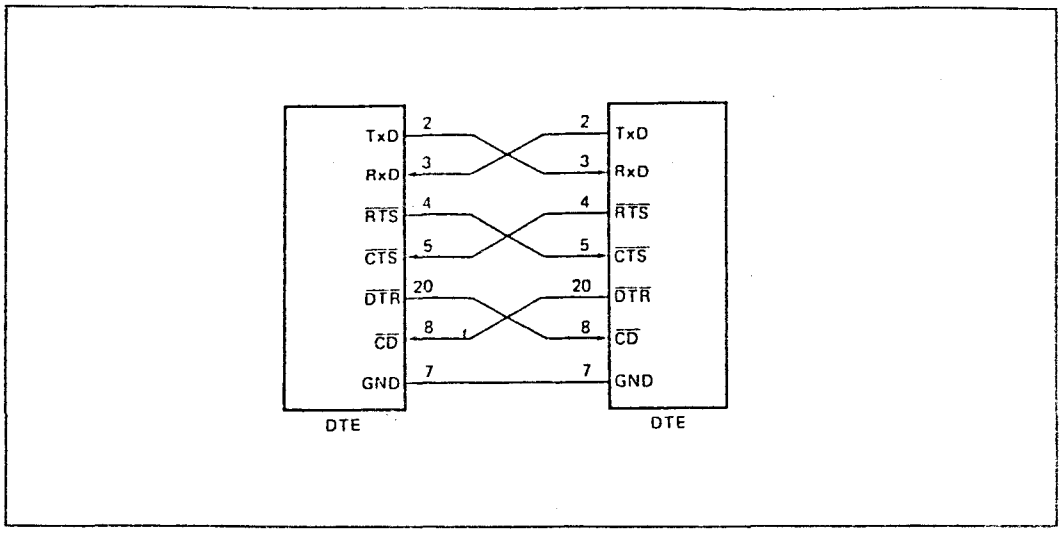
31) Hall, D. V., MICROPROCESSORS AND INTERFACING, Programming and Hardware, Mc Graw-Hill, 1985, hal. 450

32) Kruglinski, D., GUIDE TO IBM PC COMMUNICATIONS, The Osborne, Mc Graw-Hill, 1986, hal. 41

2.6.3 KONFIGURASI HUBUNGAN RS-232-C

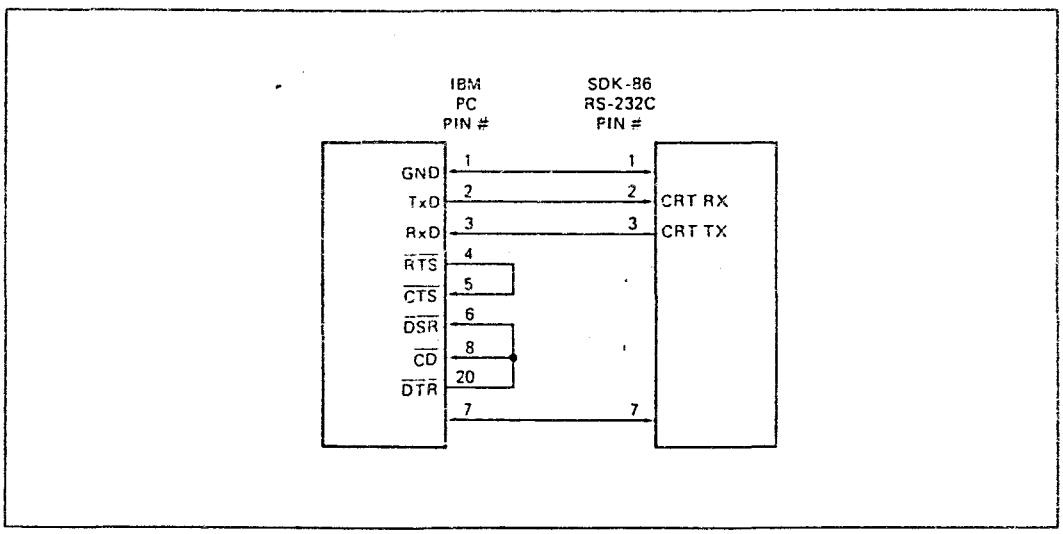
Hubungan antara dua komputer dimana jaraknya dekat, tidak memerlukan modem, namun tetap mengikuti standar RS-232-C. Hubungan ini dinamakan hubungan crossover atau sering disebut null modem seperti ditunjukkan pada gambar 2-22.

Bila kita menghubungkan komputer dengan peralatan lain, maka hubungan ini dapat menggunakan sebagian, seluruh atau tanpa sinyal handshake. Untuk hubungan yang tanpa menggunakan sinyal handshake, card IBM PC-XT dikonfigurasikan sebagai DTE. Agar card interface serial asinkron pada IBM PC-XT mampu berkomunikasi, maka input rangkaian CTS, DSR, dan CD harus diaktifkan. Bios akan mengaktifkan output dari rangkaian DTR dan RTS. Gambar 2-23 menunjukkan hubungan ini dimana pada IBM-PC, RTS pada pin 4 dihubungkan ke CTS pada pin 5, sehingga CTS akan aktif begitu RTS diaktifkan. Pin 6 dan pin 8 dan pin 20 juga dihubungkan bersama agar pada saat IBM PC-XT mengaktifkan output DTR, input DSR dan input CD akan aktif juga.



GAMBAR 2-22[33)]

HUBUNGAN RS-232-C NULL MODEM



GAMBAR 2-23[34)]

HUBUNGAN RS-232-C TANPA HANDSHAKING

---

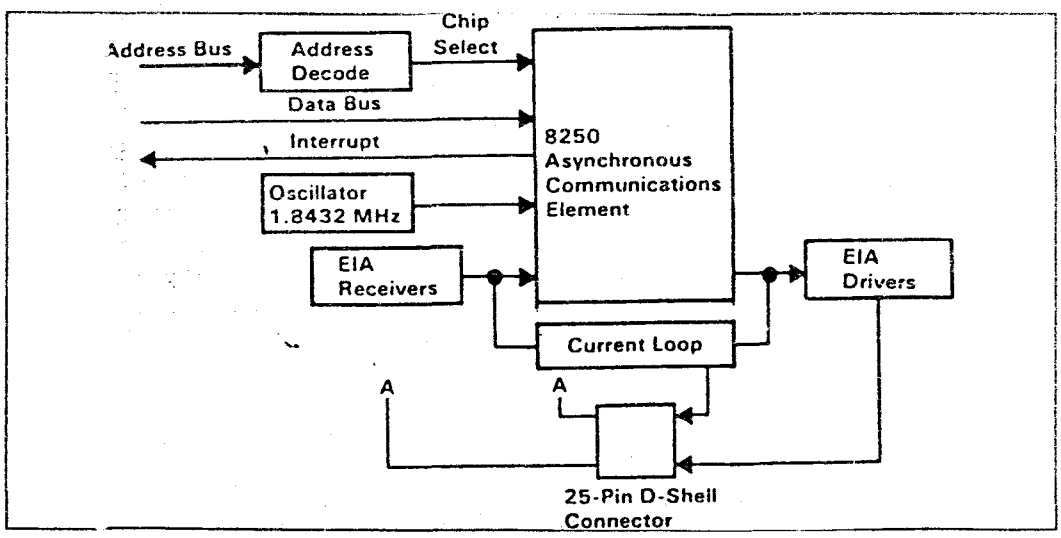
33) Ibid, hal 452

34) Hall, Douglas V., loc. cit

2.7 8250 UART

Dalam melakukan komunikasi asynchronous, pada IBM PC digunakan suatu asynchronous communication adapter. Adapter tersebut hanya digunakan untuk komunikasi asynchronous saja. Start bit, stop bit, serta parity bit. Sebuah generator baud rate yang dapat diprogram akan menyediakan operasi mulai dari 50 baud sampai dengan 9600 baud. Lima, enam, tujuh, atau delapan bit karakter dengan 1, $1^{1}/_{2}$, atau 2 stop bit juga disediakan.

Inti dari adapter tersebut adalah sebuah chip INS8250 atau ekivalennya. Gambar 2-24 sebagai berikut menunjukkan sebuah diagram kotak dari asynchronous communication adapter.



GAMBAR 2-24[35)]

DIAGRAM KOTAK ASYNCHRONOUS COMMUNICATION ADAPTER

---

35) ....., IBM Personal Computer XT Technical Reference Manual, hal. 1-186

8250 merupakan chip universal asynchronous receiver/transmitter yang mampu melakukan operasi pengiriman/penerimaan data serial dalam berbagai format data.

Chip ini mempunyai clock baud rate internal yang dapat diprogram untuk menghasilkan bermacam-macam baud rate. Selain itu 8250 juga berisi rangkaian internal yang menyebabkan pengoperasian dengan interupsi menjadi lebih mudah. 8250 memiliki 10 register 8-bit yang dapat diprogram, tetapi 10 register tersebut diakses lewat 7 port address.

Dari 10 register yang ada, hanya 6 register yang diperlukan untuk komunikasi serial yang sederhana. Transmitter holding register untuk menampung data yang baru diterima. Line control register dan line status register yang digunakan untuk menginisialisasi dan memantau 8250. Serta dua buah register lain yang penting adalah baud rate divisor (low dan high byte) yang berguna untuk menentukan baud rate. Sisa 4 register yang belum disebut adalah register untuk modem control dan modem status yang berguna untuk operasi 8250 dengan modem.

2.7.1 PEMROGRAMAN 8250

8250 mempunyai beberapa register yang dapat

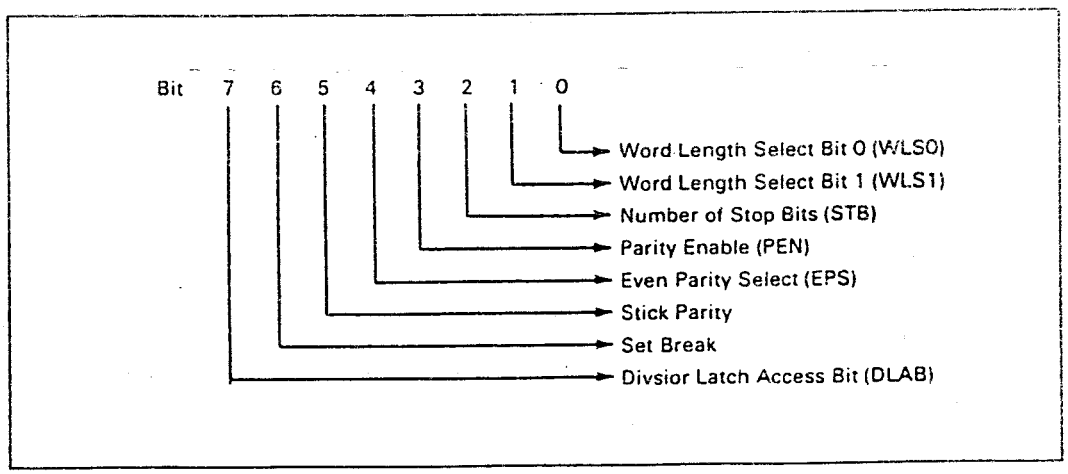
di-akses dan diprogram dengan sistem komunikasi yang diinginkan. Pemrogram dapat mengakses atau memprogram register-register 8250 melalui CPU. Register-register tersebut dapat digunakan untuk mengontrol operasi 8250, mengirim data, dan menerima data. Berikut ini adalah keterangan masing-masing register tersebut.

2.7.1.1 LINE CONTROL REGISTER (LCR)

Register ini merupakan sarana untuk memprogram format data dari sistem komunikasi serial asynchronous yang diinginkan. Isi dari Line Control Register (LCR) ditunjukkan pada gambar 2-25 berikut.

- Bit 0 dan 1 (Word Length Select Bit/ WLS0 dan WLS1)

Kedua bit ini menentukan jumlah bit atau dari setiap data karakter serial yang dikirimkan atau diterima 8250.



GAMBAR 2-25[36]

LINE CONTROL REGISTER

---

[36] Ibid, hal. 1-197

Tabel 2-4 menunjukkan kombinasi bit Ø dan 1 yang menentukan jumlah bit setiap karakter.

- Bit 2 (Number of Stop Bit/STB)

Bit ini menentukan jumlah stop bit dari setiap data karakter yang dikirim atau diterima 8250. Jika bit 2

TABEL 2-4

KOMBINASI BIT 1 DAN BIT 0 DARI LCR

bit 1	bit 2	word length
Ø	Ø	5 bit
Ø	1	6 bit
1	Ø	7 bit
1	1	8 bit

berlogika 'Ø', maka jumlah stop bit adalah 1. Jika bit 2 berlogika '1' dan panjang setiap data karakter 5 bit maka jumlah stop bit adalah $1^{1}/_{2}$. Jika bit 2 berlogika '1' tetapi panjang data karakter 6, 7 atau 8 bit maka stop bit berjumlah 2.

- Bit 3 (Parity Enable/PEN)

Bit ini merupakan parity enable bit dimana logika '1' pada bit ini akan menyebabkan bit parity dibangkitkan (pada sisi kirim) atau dideteksi (pada sisi terima). Bit parity ini digunakan untuk menghasilkan jumlah '1' genap

(even parity) atau ganjil (odd parity) bilamana bit-bit '1' dari data karakter dan bit parity dihitung banyaknya.

- Bit 4 (Even Parity Select)

Bit ini digunakan untuk memilih parity genap (even parity) atau parity ganjil (odd parity). Logika '1' pada bit 3 (PEN) dan logika 'Ø' pada bit 4 ini merupakan parity ganjil dan logika '1' pada bit 3 (PEN) dan logika '1' pada bit 4 merupakan parity genap.

- Bit 5 (Stick Parity)

Logika '1' pada bit 5 ini dan logika '1' pada bit 3 (PEN) akan menyebabkan parity bit dikirimkan dan kemudian dideteksi oleh penerima sebagai logika 'Ø' bila bit 4 berlogika '1' atau sebagai logika '1' apabila bit 4 berlogika 'Ø'.

- Bit 6 (Set Break)

Logika '1' pada bit ini menyebabkan serial output (SOUT) berada pada kondisi SPACING (logika 'Ø') dan tetap demikian walaupun bagian transmisi masih bekerja. Set Break ini dapat dimatikan (disabled) dengan menge-set bit 6 pada logika Ø.

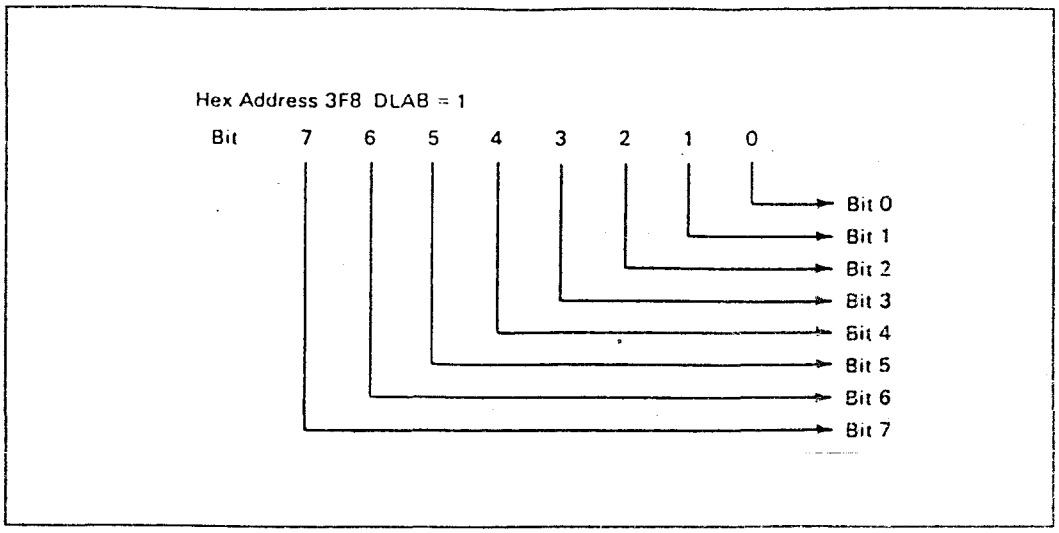
- Bit 7 (Divisor Latch Access Bit/DLAB)

Logika '1' pada bit ini menyebabkan CPU dapat mengakses divisor latch dari pembangkit baud rate selama operasi read atau write. Ketika CPU mengakses receiver buffer,

transmitter holding register atau interrupt enable register, bit 7 ini harus berlogika '0'.

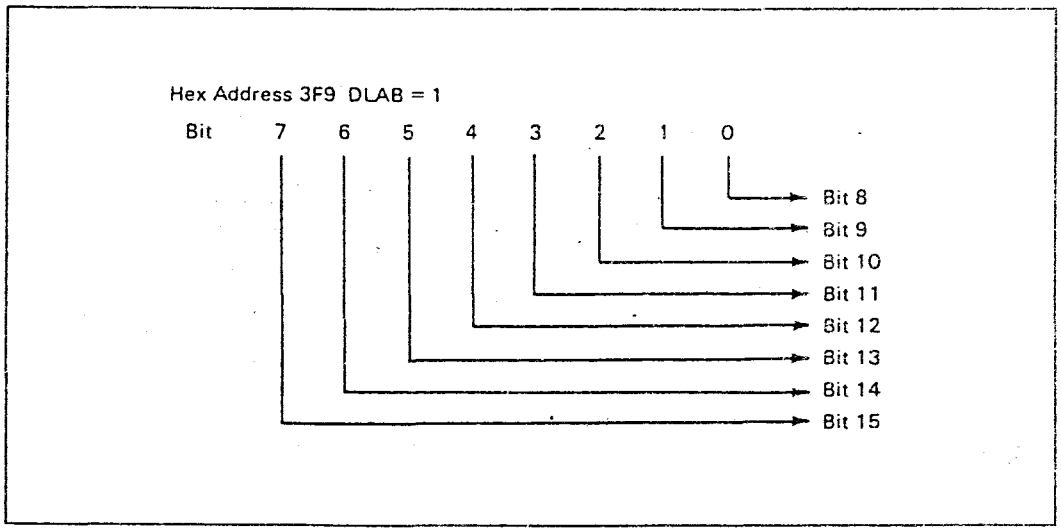
2.7.1.2 DIVISOR LATCH LEAST/MOST SIGNIFICANT BIT (DLL DAN DLM)

8250 berisi programable baud rate generator yang mampu membagi clock input dengan suatu pembagi dari 1 sampai (2^{16}-1). Frekuensi output dari baud generator sebesar 16 x baud rate (pembagi= frekuensi clock input/(16 x baud rate)). Dua register latch 8 bit digunakan untuk menyimpan pembagi dalam format 16 bit binner. Divisor register latch ini harus diakses selama proses inisialisasi agar operasi dari baud rate generator sesuai dengan yang dikirimkan. Gambar 2-26 dan 2-27 menunjukkan konfigurasi dari 16 bit register latch. Frekuensi maksimum yang diperbolehkan pada baud rate generator 8250 adalah sebesar 3.1 MHz. Tabel 2-5 menunjukkan angka-angka pembagi yang digunakan untuk menghasilkan macam-macam baud rate generator sebesar 2 MHz.



GAMBAR 2-26[37)]

DIVISOR LATCH LEAST SIGNIFICANT BIT (DLL)



GAMBAR 2-27[38)]

DIVISOR LATCH MOST SIGNIFICANT BIT (DLM)

---

37) Ibid, hal. 1-199

38) Ibid, hal. 1-200

TABEL 2-5[39)]

ANGKA-ANGKA PEMBAGI PADA FREKUENSI CLOCK 2 MHz

Baud rate yg diinginkan	Besar pembagi yang digunakan untuk menghasilkah 16 X clock	
	desimal	heksa
50	2500	09CA
75	1666	0682
110	1136	0470
150	833	0341
300	416	01A0
600	208	00D0
1200	104	0068
2400	52	0034
4800	26	001A
7200	17	0011
9600	13	000D

2.7.1.3 LINE STATUS REGISTER (LSR)

Register 8 bit ini memberikan informasi tentang status dari CPU yang berkaitan dengan transfer data. Isi line status register ditunjukkan pada gambar 2-28.

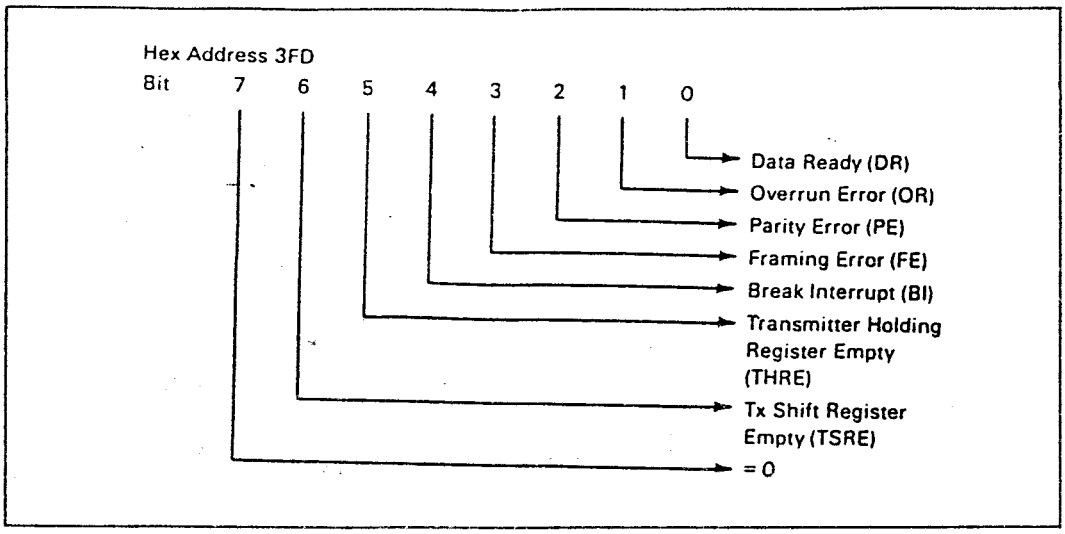
- Bit 0 (Data Ready/DR)

Bit ini merupakan indikator dari receiver data ready (DR). Bit 0 ini akan di-set '1' bilamana karakter yang diterima sudah lengkap dan siap dikirimkan ke receiver buffer register. Bit 0 dapat di-reset berlogika '0' baik pada saat CPU sedang membaca data pada receiver buffer register atau ketika CPU menulis '0' pada register ini.

- Bit 1 (Overrun Error/OE)

Bit ini merupakan indikasi adanya overrun error. Overrun

[39)] Ibid, hal. 1-200



GAMBAR 2-28[40]

LINE STATUS REGISTER

error ini terjadi jika data yang ada pada receiver buffer register belum sempat terbaca oleh CPU namun receiver buffer register sudah diisi lagi dengan data yang baru sehingga data yang belum sempat terbaca hilang. Bit OE akan di-reset ketika CPU membaca isi line status register.

- Bit 2 (Parity Error/PE)

Bit ini merupakan indikasi adanya parity error. Parity error ini terjadi bilamana data yang diterima tidak mempunyai jumlah parity yang tepat seperti ketika di-set pertama kali (genap/ganjil). Bit ini akan berlogika '1' ketika terjadi parity error dan di-reset saat line register dibaca oleh CPU.

---

[40] Ibid, hal. 1-201

- Bit 3 (Framming Error/FE)

Logika '1' pada bit ini menunjukkan bahwa framming error terjadi. Framming error terjadi jika karakter yang diterima tidak mempunyai stop bit yang tepat. Bit ini di-reset pada saat line register dibaca oleh CPU.

- Bit 4 (Break Interrupt)

Bit ini merupakan indikator terjadinya break interrupt. Bit ini di-set '1' bilamana data yang diterima berlogika 'Ø' selama lebih dari waktu yang dibutuhkan untuk 1 data karakter (total waktu start bit + data bit + parity + stop bit). Bit ini di-reset ketika CPU membaca line status register.

- Bit 5 (Transmitter Holding Register Empty/THRE)

Bit ini menunjukkan bahwa 825Ø siap menerima data karakter baru yang akan dikirim. Aktifnya bit ini dapat menyebabkan 825Ø menginterrupt CPU bilamana Transmitter Holding Register Empty Interrupt Enable di-set '1'. Bila THRE di-set '1' bila data karakter sudah ditransfer ke Transmitter Shift Register dan di-reset saat Transmitter Holding Register dibaca CPU.

- Bit 6 (Transmitter Shift Register Empty/TSRE)

Logika '1' pada bit ini menandakan bahwa Transmitter Shift Register sedang menunggu adanya karakter dari Transmitter Holding Register. Bit ini di-reset pada saat Transmitter Holding Register mengirim data ke

Transmitter Shift Register. Bit 6 ini merupakan bit yang hanya dapat dibaca.

- Bit 7

Bit ini selalu di-reset '0'.

2.7.1.4 INTERRUPT IDENTIFICATION REGISTER

8250 merupakan rangkaian interrupt internal yang dapat dikendalikan/diprogram dengan perangkat lunak. Selain itu interrupt pada 8250 ini juga dilengkapi dengan prioritas 4 tingkat (level) dengan urutan sebagai berikut :

Prioritas 1: Receiver Line Status

Prioritas 2: Received Data Ready

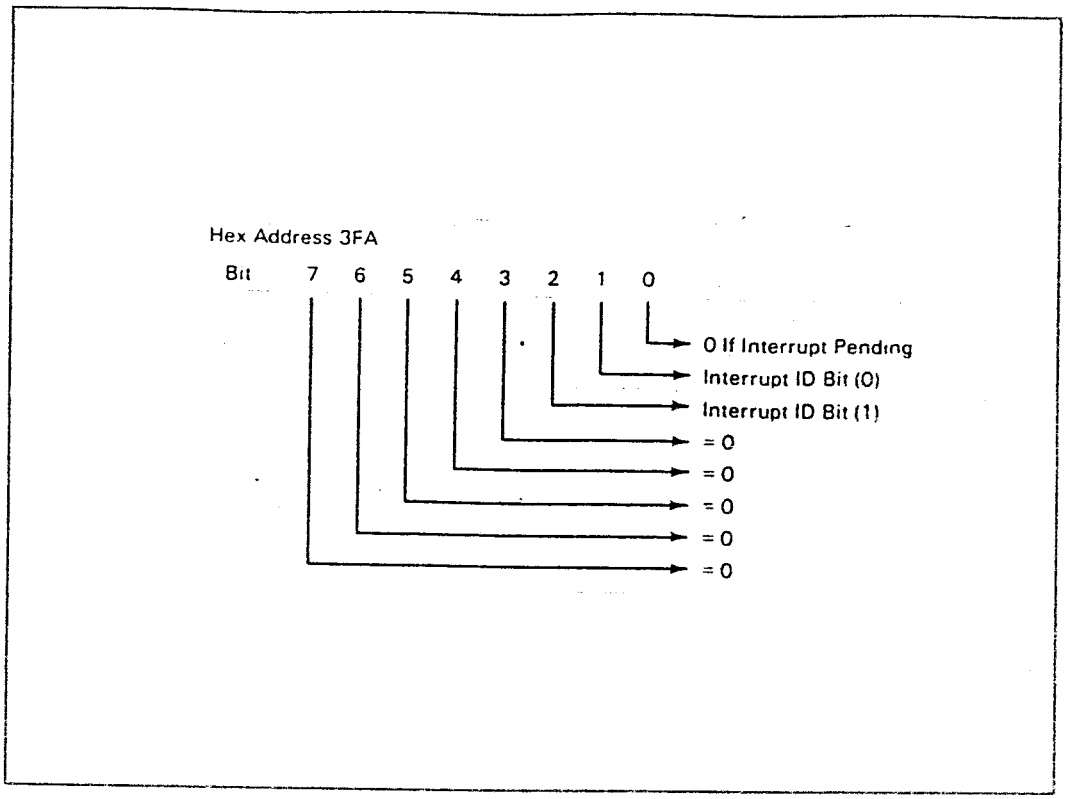
Prioritas 3: Transmitter Holding Register Empty

Prioritas 4: Modem Status

informasi tentang interrupt tersebut selalu dideteksi dan tipe dari prioritas interrupt disimpan pada interrupt identification register ditunjukkan pada gambar 2-29.

- Bit 0 (Interrupt Pending)

Logika '0' pada bit ini menunjukkan bahwa kondisi interrupt terjadi. Dan logika '1' pada bit ini menunjukkan bahwa interrupt tidak terjadi dan proses polling tetap berlanjut.



GAMBAR 2-29[41]

INTERRUPT IDENTIFICATION REGISTER

- Bit 1 dan 2 (Interrupt ID)

Dua bit ini digunakan untuk menentukan prioritas interrupt yang akan terjadi. Tabel 2-6 menunjukkan kombinasi bit-bit Ø,1 dan 2 dari interrupt identification register menentukan interrupt control Function.

- Bit 3 sampai dengan 7

Bit 3 sampai dengan bit 7 selalu di-set 'Ø'.

---

[41] Ibid, hal. 1-203

TABEL 2-6[42)]

FUNGSI - FUNGSI DARI INTERRUPT CONTROL

Interrupt ID Register				Interrupt Set and Reset Functions		
Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	1		None	None	
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	Modem Status	Clear to Send or Data Set Ready or Ring Indicator or Received Line Signal Direct	Reading the Modem Status Register

2.7.1.5 INTERRUPT ENABLE REGISTER

Register 8 bit ini memungkinkan keempat bentuk interrupt yang ada pada 8250 untuk secara terpisah mengaktifkan sinyal output INTRPT. Selain itu lewat

---

[42)] Ibid, hal. 1-204

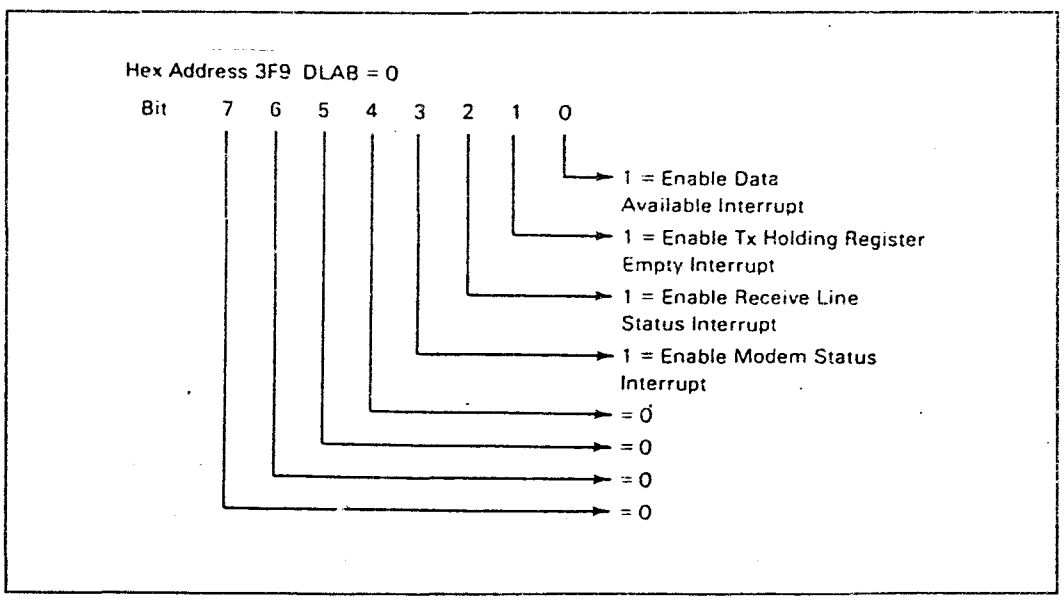
register ini sistem interrupt yang ada juga dapat dimatikan dengan cara me-reset bit Ø sampai bit 3 pada logika 'Ø'. Keempat bentuk interrupt yang ada pada 8250 masing-masing dapat diaktifkan dengan cara menge-set bit Ø sampai bit 3 yang sesuai dengan bentuk interrupt yang dikehendaki. Isi dari interrupt enable register ditunjukkan pada gambar 2-30.

- Bit 0

Logika '1' pada bit ini akan mengaktifkan bentuk interrupt received data ready (prioritas 2).

- Bit 1

Logika '1' pada bit ini akan mengaktifkan bentuk interrupt transmitter holding register empty.



GAMBAR 2-30[43)]

INTERRUPT ENABLE REGISTER

[43)] Ibid, hal. 1-205

- Bit 2

Logika '1' pada bit ini akan mengaktifkan bentuk interrupt receive line status (prioritas 1).

- Bit 3

Logika '1' pada bit ini akan mengaktifkan bentuk interrupt modem status (prioritas 4).

- Bit 4 s/d bit 7

Keempat bit ini selalu di-set 'Ø'.

2.7.1.6 MODEM CONTROL REGISTER

Register ini digunakan untuk mengontrol modem. Isi dari modem control register seperti pada gambar 2-31.

- Bit 0 (Data Terminal Ready)

Bit ini digunakan untuk mengontrol sinyal output data terminal ready. Logika '1' pada bit ini akan menyebabkan pin $\overline{DTR}$ berlogika 'Ø'. Sebaliknya logika 'Ø' pada bit ini akan menge-set pin $\overline{DTR}$ berlogika 1.

- Bit 1 (Request To Send)

Merupakan bit pengontrol pin output request to send ($\overline{RTS}$). Keadaan bit 1 ini dalam mempengaruhi pin $\overline{RTS}$ sama seperti pada bit Ø.

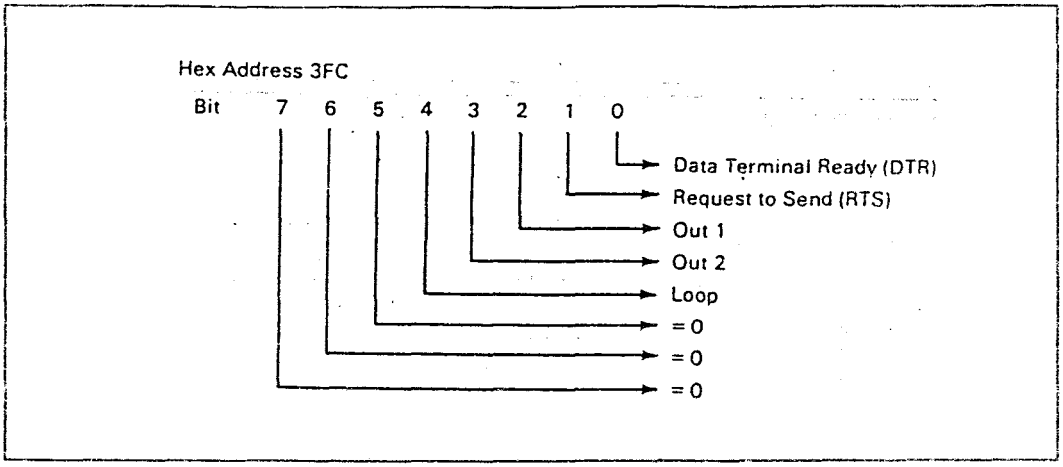
- Bit 2 (OUT 1)

Logika '1' pada bit ini menyebabkan pin $\overline{OUT\ 1}$ pada kondisi 'Ø' dan sebaliknya logika 'Ø' akan menyebabkan

$\overline{\text{OUT 1}}$ pada kondisi '1'.

- Bit 3 (OUT 2)

Logika '1' pada bit ini menyebabkan pin $\overline{\text{OUT 2}}$ pada



GAMBAR 2-31[44)]

MODEM CONTROL REGISTER

kondisi '0' dan sebaliknya logika '0' akan menyebabkan $\overline{\text{OUT 2}}$ pada kondisi '1'.

- Bit 4 (Loop)

Bit ini merupakan sarana untuk memberikan operasi dari 8250. Logika '1' pada bit ini akan menyebabkan hal-hal sebagai berikut:

- Transmitter Serial Output (SOUT) berada pada disi 'marking' dan Receiver Serial Input (SIN) tidak terhubung.
- Out dari Transmitter Shift Register diumpankan kembali ke input Receiver Shift Register.

[44)] Ibid, hal. 1-206

- Keempat input pengontrol modem ($\overline{CTS}$, $\overline{DSR}$, $\overline{RLSD}$, dan $\overline{RI}$) tidak terhubung dan keempat output pengontrol modem ($\overline{DTR}$, $\overline{RTS}$, $\overline{OUT\ 1}$, dan $\overline{OUT\ 2}$) dihubungkan internal ke input dari keempat input pengontrol modem di atas. Pada saat pengujian dilakukan, data yang dikirimkan segera diterima kembali. Keistimewaan ini membuat CPU dapat memeriksa pengiriman dan penerimaan data pada 8250. Ketika pengujian berlangsung, sistem interrupt dapat diaktifkan dengan interrupt enable register.

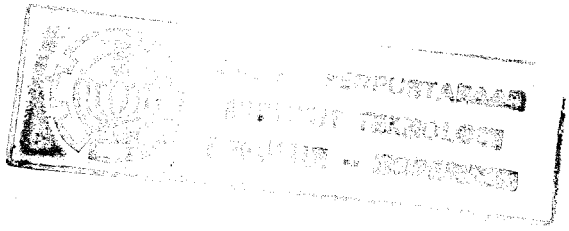
Logika '0' pada bit ini akan mengembalikan 8250 pada operasi normal.

- Bit 5 s/d bit 7

Ketiga bit ini selalu pada logika '0'.

2.7.1.7 MODEM STATUS REGISTER

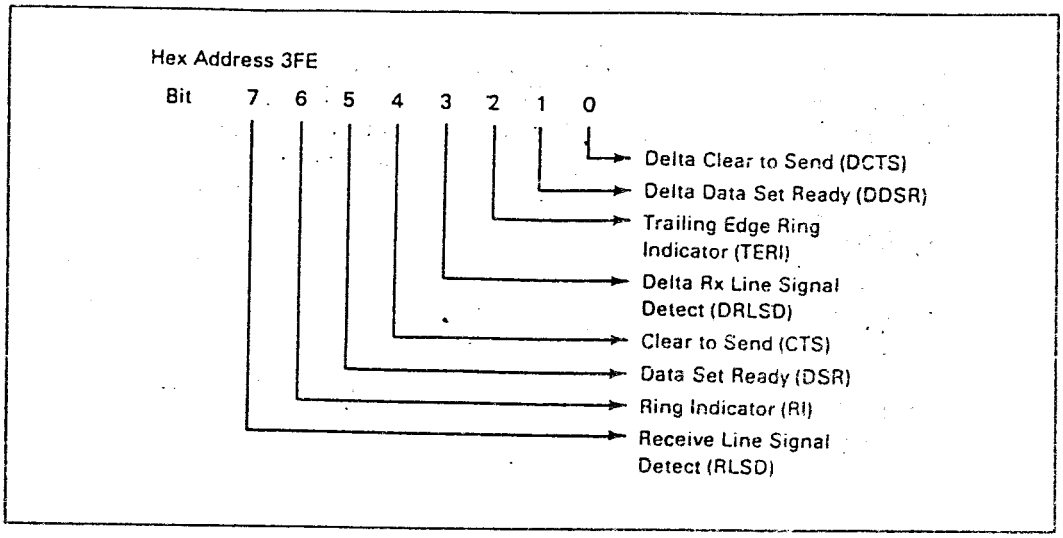
Merupakan 8 bit register yang digunakan sebagai indikator dari pin-pin pengontrol modem. Empat bit dari modem status register ini digunakan untuk memberikan informasi tentang perubahan-perubahan yang terjadi pada pin-pin pengontrol modem. Bit-bit ini akan di-set '1' bilamana ada perubahan yang terjadi dan akan di-reset '0' pada saat CPU membaca modem status register ini. Bit-bit dari modem status register ini. Bit-bit dari



modem status register ditunjukkan pada gambar 2-32.

- Bit 0 (Delta Clear To Send/DCTS)

Merupakan bit indikator dari Delta Clear To Send yang



GAMBAR 2-32[45)]

MODEM STATUS REGISTER

menunjukkan bahwa input $\overline{CTS}$ telah berubah sejak terakhir kali ketika dibaca CPU.

- Bit 1 (Delta Data Set Ready/DDSR)

Merupakan bit indikator Delta Data Set Ready yang menunjukkan bahwa input $\overline{DSR}$ telah berubah sejak terakhir kali ketika dibaca CPU.

- Bit 2 (Trailing Edge Ring Indikator/TERI)

Merupakan bit indikator dari Trailing Edge Ring yang menunjukkan input $\overline{RI}$ telah berubah dari logika '1' menuju logika 'Ø'.

[45)] Ibid, hal. 1-208

- Bit 3 (Delta RX Line Signal Detect)

Merupakan bit indikator dari Delta Received Line Signal Detector yang menunjukkan bahwa input $\overline{RLSD}$ telah berubah keadaan.

- Bit 4 (Clear To Send/CTS)

Bit ini merupakan komplemen sinyal input CTS. Bila bit 4 (loop) dari MCR pada logika '1', bit ini merupakan RTS pada MCR.

- Bit 5 (Data Set Ready/DSR)

Bit ini merupakan komplemen dari sinyal input DSR. Bila bit 4 (loop) dari MCR pada logika '1', bit ini merupakan DTR pada MCR.

- Bit 6 (Ring Indicator/RI)

Bit ini merupakan komplemen dari sinyal input $\overline{RI}$. Bila bit 4 (loop) dari MCR pada logika '1', bit ini merupakan OUT 1 pada MCR.

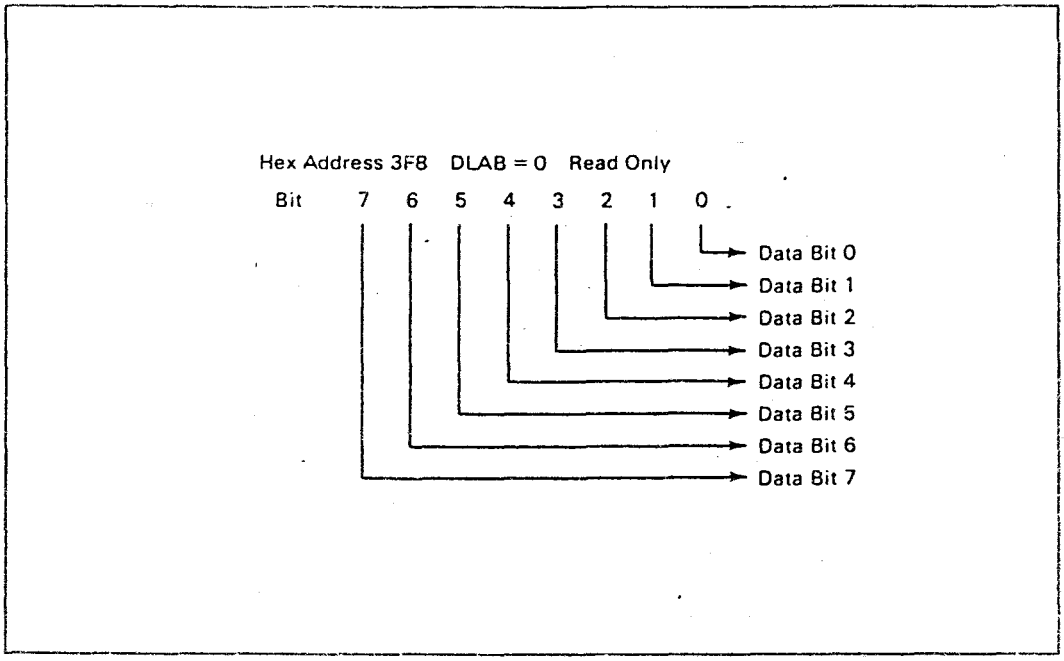
- Bit 7 (Receiver Line Signal Detect/RLSD)

Bit ini merupakan komplemen dari sinyal input RLSD. Bila bit 4 (loop) dari MCR pada logika '1', bit ini merupakan OUT 2 pada MCR.

2.7.1.8 RECEIVER BUFFER REGISTER

Receiver buffer register berisi data karakter yang diterima. Bit Ø merupakan Least Significant Bit dan

pertama kali diterima. Receiver Buffer Register ditunjukkan pada gambar 2-33.



GAMBAR 2-33[46]

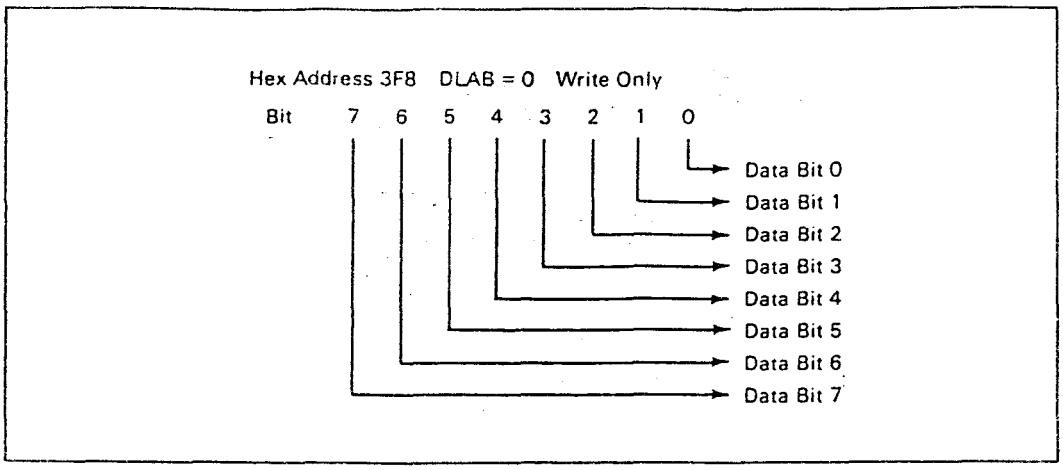
RECEIVER BUFFER REGISTER

2.7.1.9 TRANSMITTER HOLDING REGISTER

Transmitter Holding Register berisi data karakter yang akan ditransmisikan secara serial. Bit Ø merupakan Least Significant Bit dan ditransmisikan pertama kali. Transmitter Holding Register ditunjukkan pada gambar 2-34.

---

[46] Ibid, hal. 1-210



GAMBAR 2-34[47)]

TRANSMITTER HOLDING REGISTER

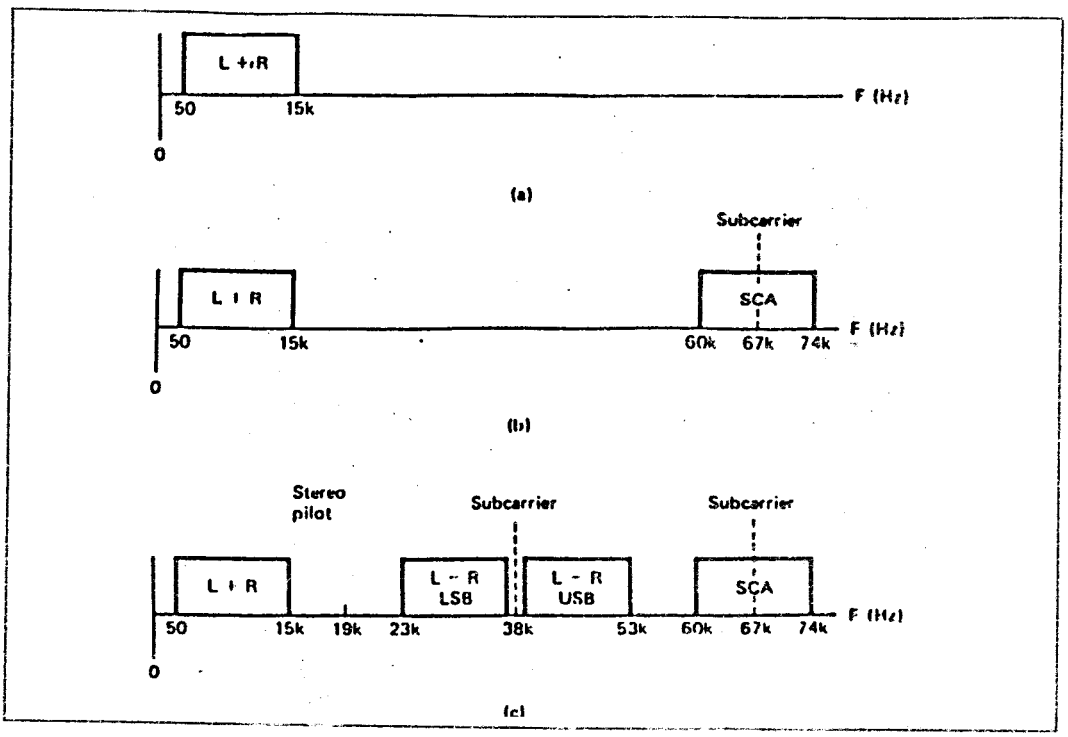
2.8. MODULASI FREKUENSI STEREO

Sampai dengan tahun 1961, semua sistem pemancar FM komersial adalah *monophonic*, dimana hanya dipakai satu saluran (channel) untuk audio antara 50 Hz sampai dengan 15 kHz yang terdiri dari informasi suara manusia dan musik. Sedangkan untuk saluran frekuensi pembawa (*carrier*) ditetapkan sebesar 200 kHz. Dengan sistem ini pada speaker akan dihasilkan informasi yang sama, dimana suara akan berasal dari satu titik. Akan tetapi pada sistem ini dapat juga dipisahkan suara berdasar pada frekuensi (*frequency domain*), yaitu : dengan menggunakan *woofer* frekuensi rendah dan *tweeter* untuk frekuensi tinggi. Pada tahun 1955 FCC menemukan transmisi *subcarrier* untuk *Subsidiary Communication*

[47)] Ibid, hal. 1-211

Authorization (SCA). SCA ini digunakan untuk penyiaran musik kepada pelanggan tanpa bisa diputus, misalnya restauran, supermarket, dan sebagainya. *Subcarrier* SCA terletak pada 67 kHz. Sinyal *subcarrier* dan sinyal-sinyal *sideband* dimultiplekskan dengan sinyal pembawa menggunakan teknik FDM (*Frequency Division Multiplexing*). *Subcarrier* SCA dapat berupa AM double sideband atau single sideband dan terletak antara 60 sampai dengan 74 kHz. Sehingga frekuensi maksimum sinyal pemodulasi adalah 7 kHz, untuk itu digunakan narrowband FM. Deviasi frekuensi total adalah 75 kHz, dimana 90% (67,5 kHZ) dialokasikan untuk saluran utama, sedangkan 10% (7,5 kHz) dialokasikan untuk SCA.

Pada tahun 1961 saluran audio yang asli (5Hz-15kHz) ditambah dengan dua saluran lagi dengan wilayah frekuensi yang berbeda, yaitu : saluran sinyal audio kiri dikurangi sinyal audio kanan (L-R) yang menduduki daerah 23 kHz sampai dengan 53 kHz, dan sinyal subcarrier SCA dan sideband-nya pada daerah 60 kHz sampai dengan 74 kHz. Sistem ini mempunyai deviasi frekuensi sebesar 75 kHz, dimana 10% (7,5 kHz) dipakai untuk transmisi SCA dan 10% yang lain dipakai untuk *stereo pilot*. Gambar 2-35 menjelaskan tentang spektrum baseband FM.



GAMBAR 2-35[48)]

SPEKTRUM BASEBAND FM A) SAMPAI DENGAN TH.1951

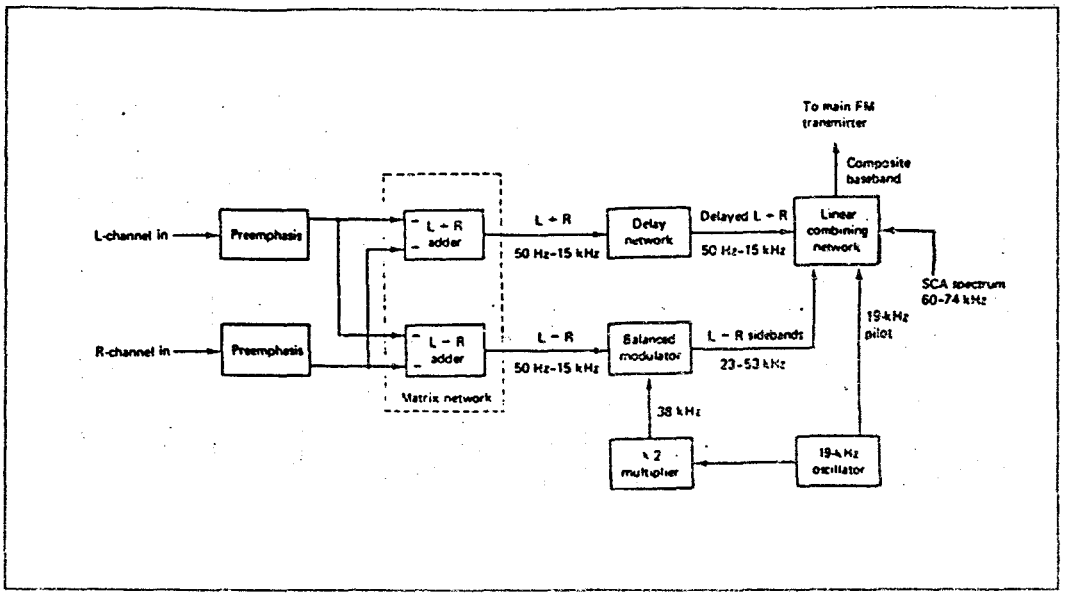
B) SAMPAI DENGAN TH. 1961 C) SEJAK TH. 1961

2.8.1 PEMBANGKITAN FM STEREO

Pada gambar 2-36 terlihat blok diagram dari pembangkit FM stereo. Dari gambar itu nampak bahwa sinyal L+R dan L-R didapat dari rangkaian matriks. Sinyal L-R ini kemudian memodulasi sinyal pembawa 38 kHz dengan metode AM DSBSC, sedangkan sinyal L+R di-delay untuk menjaga integritas fase dengan sinyal L-R. Kemudian sinyal pilot 19 kHz yang merupakan subharmonisa

---

48) Tomasi, W., FUNDAMENTALS OF ELECTRONIC COMMUNICATIONS SYSTEMS, Prentice Hall, Englewood Cliffs, New Jersey, 1988, hal. 341



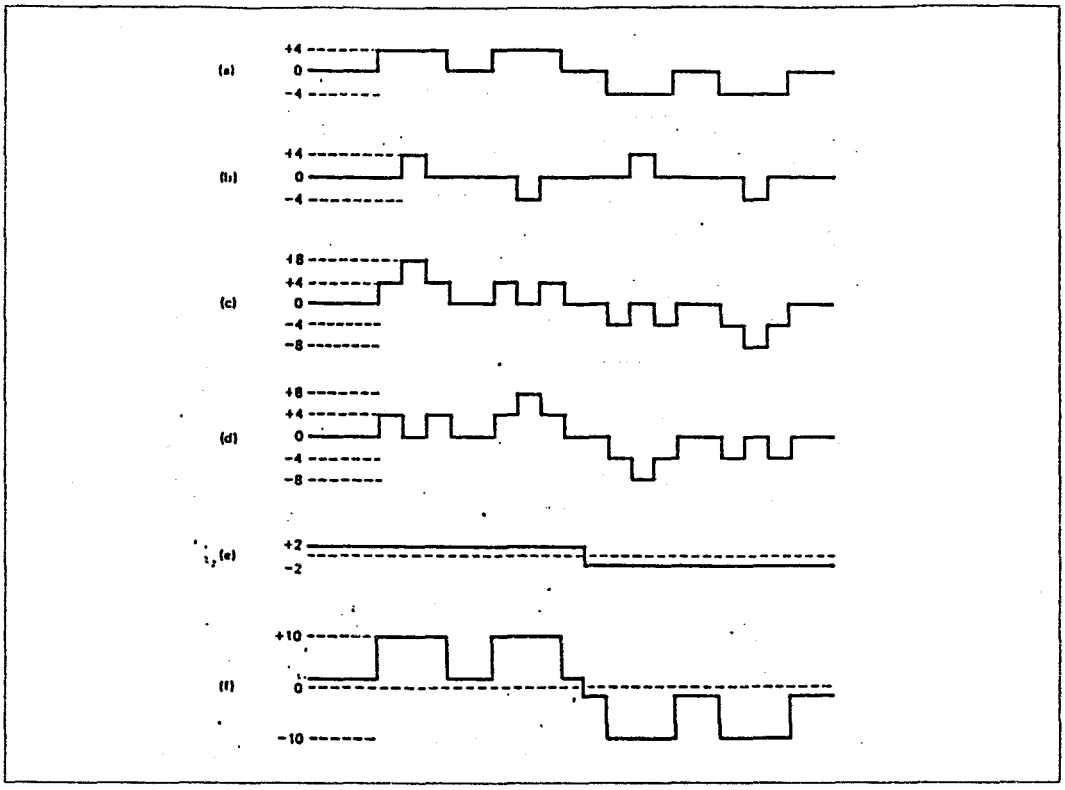
GAMBAR 2-36[49]

PEMANCAR FM-STEREO MENGGUNAKAN TEKNIK FDM

dari 38 kHz ditambahkan ke baseband untuk kemudian dimodulasikan ke sinyal pembawa. Sistem multipleks yang dipakai adalah FDM (*Frequency Division Multiplexing*) yaitu dengan menjumlahkan amplitudo sinyal-sinyal tersebut dengan basis frekuensi yang berbeda seperti terlihat pada gambar 2-37 dan tabel 2-6 untuk amplitudo sinyal yang sama. Sedangkan untuk amplitudo sinyal yang berbeda dapat dilihat pada gambar 2-38 Nampak bahwa jumlah sinyal-sinyal tersebut (sinyal composit) besarnyal tidak melebihi 1Ø V, dengan demikian deviasi frekuensinya tidak melebihi 75 kHz. Hal ini sesuai dengan peraturan yang telah ditetapkan oleh FCC.

---

[49] Ibid, hal. 343



GAMBAR 2-37[50)]

MULTIPLEKS SINYAL STEREO UNTUK AMPLITUDO YANG SAMA

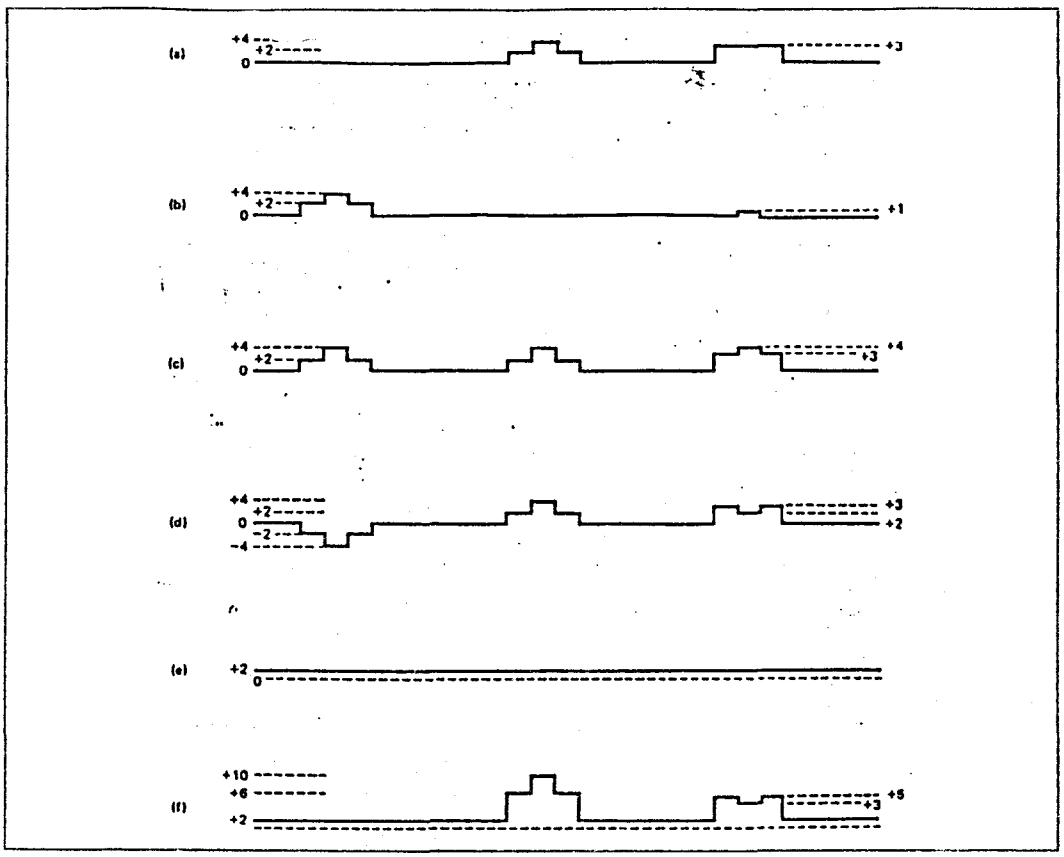
TABEL 2-7[51)]

TEGANGAN SINYAL COMPOSIT FM

L	R	L+R	L-R	SCA&PILOT	TOTAL
Ø	Ø	Ø	Ø	2	2
4	Ø	4	4	2	1Ø
Ø	4	4	4	2	2
4	4	8	Ø	2	1Ø
4	-4	Ø	8	2	1Ø
-4	4	Ø	-8	2	-1Ø
-4	-4	-8	Ø	2	-1Ø

50) Ibid, hal. 344

51) Ibid, hal. 345



GAMBAR 2-38[52)]

MULTIPLEKS SINYAL STEREO UNTUK AMPLITUDO YANG BERBEDA

2.8.2 PENERIMA FM STEREO

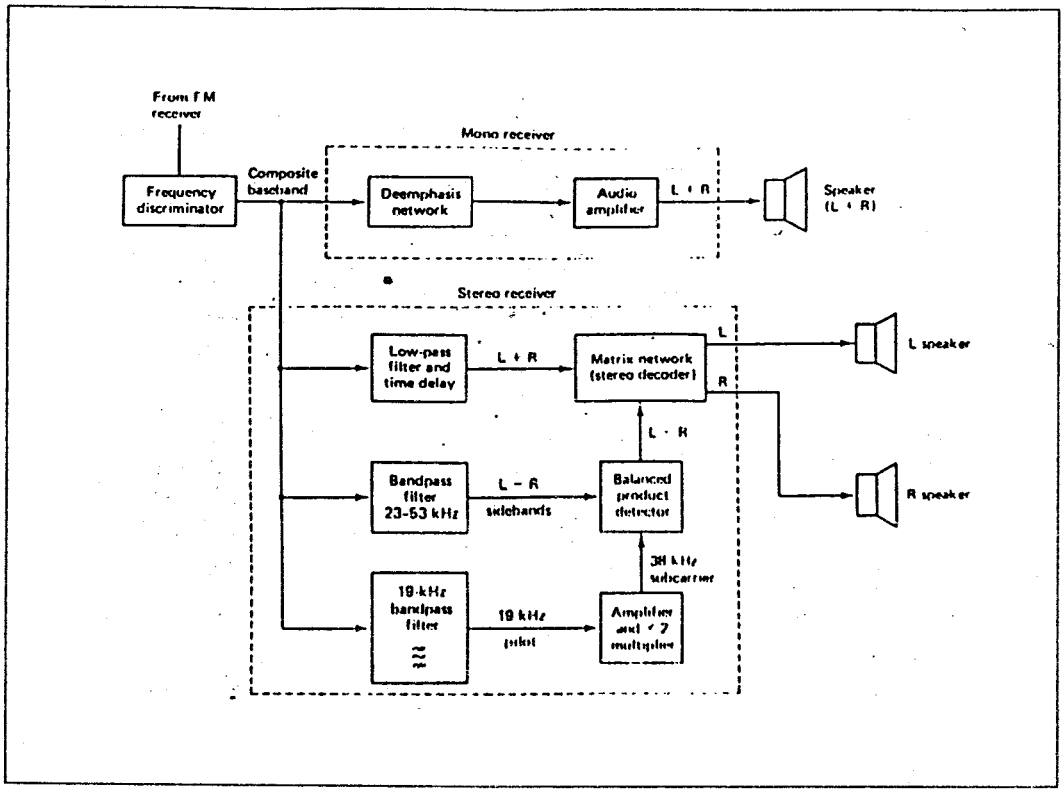
Sistem penerima FM stereo identik dengan penerima FM mono, hanya terdapat perbedaan sedikit pada bagian audionya. Pada sistem mono, sinyal L+R setelah difilter dan dikuatkan langsung dioutputkan ke speaker. Sedangkan pada pemrosesan sistem stereo, sinyal L+R dipisahkan dari sinyal komposit dengan filter LPF, sedang sinyal L-R double-sideband dipisahkan dengan filter BPF, dan

---

[52)] Ibid, hal. 346

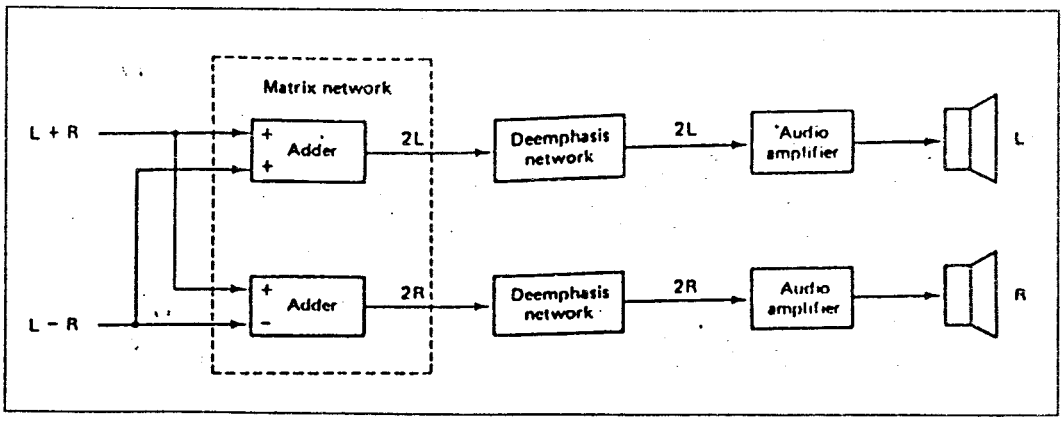
juga sinyal pilot 19 kHz dipisahkan dengan filter BPF dengan faktor kwalitas yang tinggi. Selanjutnya sinyal pilot 19 kHz dikalikan 2 menjadi 38 kHz untuk kemudian dicampur dengan sinyal L-R pada *Balance Product Detector*. Pada akhirnya sinyal L+R dan sinyal L-R diinputkan ke rangkaian matriks untuk dipisahkan menjadi sinyal L dan sinyal R dan dioutputkan ke speaker. Gambar 2-39 menjelaskan bagaimana sistem penerima FM stereo ini bekerja. Sedangkan gambar 2-4Ø menjelaskan mengenai rangkaian dekoder matriks stereo.

Selain dengan metode ini, pada saat ini banyak dipakai piranti single chip yang berfungsi sebagai demodulator FM stereo. Sebagai contoh dipilih IC XR-131Ø stereo demodulator, IC ini menggunakan teknik *phase lock* untuk memisahkan sinyal kanan (R) dan sinyal kiri (L) dari sinyal komposit. IC ini tidak membutuhkan komponen eksternal untuk rangkaian tanki. IC ini tidak membutuhkan pengaturan yang kritis, tetapi mempunyai kemampuan pemisahan saluran yang sangat bagus, distorsi yang rendah, dan jangkauan dinamis (*dynamic range*) yang luas.



GAMBAR 2-39[53)]

SISTEM PENERIMA FM MONO DAN STEREO



GAMBAR 2-40[54)]

RANGKAIAN DECODER MATRIKS STEREO

---

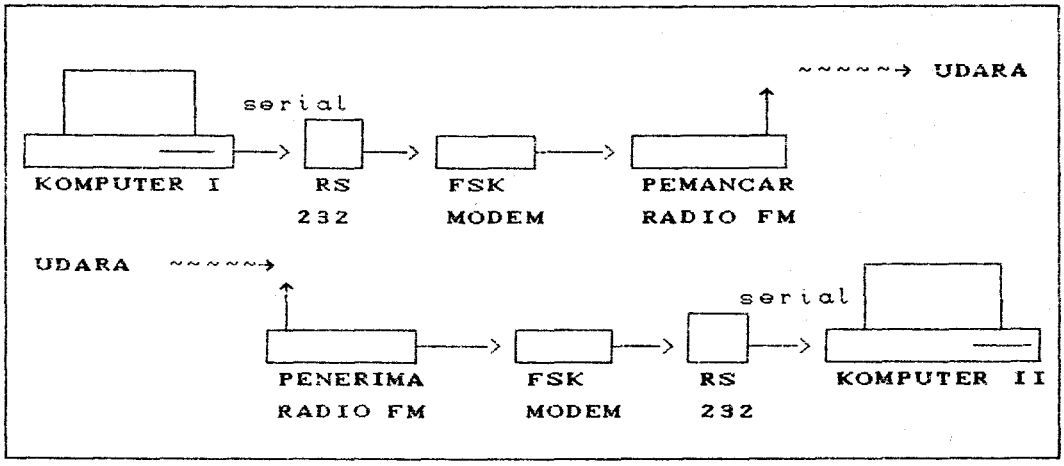
53) Ibid, hal. 347

54) Ibid, hal. 348

BAB III

PERENCANAAN DAN PEMBUATAN

Pada bab ini akan dibahas perencanaan perangkat keras sistem, yang meliputi dua hal yaitu sistem pemancar radio FM-Stereo dan sistem modulasi FSK, serta dibahas pula mengenai perangkat lunak pendukung sistem komunikasi antar komputer. Pada gambar 3-1 dijelaskan diagram blok dari hubungan sistem yang dibuat.



GAMBAR 3-1

DIAGRAM BLOK HUBUNGAN SISTEM

3.1 PERENCANAAN MODEM FSK

Modem merupakan singkatan dari modulator dan demodulator. Modem berfungsi untuk merubah data binner (sinyal digital) menjadi sinyal analog dan sebaliknya yang dalam hal ini menggunakan teknik *Frequency Shift Keying*.

3.1.1 MODULATOR

Sebagaimana telah disebutkan, bahwa pada komunikasi data melalui pemancar, sinyal-sinyal digital akan diubah ke bentuk analog dan sebaliknya, dimana proses pengubahan ini disebut modulator. Suatu modulator yang memenuhi syarat adalah yang sesuai dengan ketentuan-ketentuan yang berlaku padanya, yaitu :

- Kestabilan frekuensi.
- Kestabilan amplitudo carrier.
- Kontinyuitas fasa pada waktu transisi dari sinyal 'mark' ke 'space' atau sebaliknya.
- Kestabilan menghadapi suhu yang tinggi.

Adapun modulator yang akan dipakai pada sistem ini adalah yang menggunakan sistem modulasi *Frequency Shift Keying*, dengan pertimbangan modulasi FSK ini mempunyai kekebalan yang relatif tinggi terhadap gangguan-gangguan oleh noise atau yang lainnya. Modulator yang dipakai di sini mempunyai kecepatan pengiriman data maksimum

sebesar 300 baud.

3.1.2 DEMODULATOR

Fungsi dari demodulator adalah untuk mengubah sinyal-sinyal analog yang diterima menjadi sinyal-sinyal digital. Mengingat fungsi dari demodulator yang sedemikian pentingnya, maka harus memenuhi persyaratan-persayaratan sebagai berikut :

- Mempunyai sensitivitas yang tinggi.
- Mampu menekan input level noise yang timbul.
- Mempunyai fasilitas carrier detect.
- Kestabilan pada suhu yang tinggi.

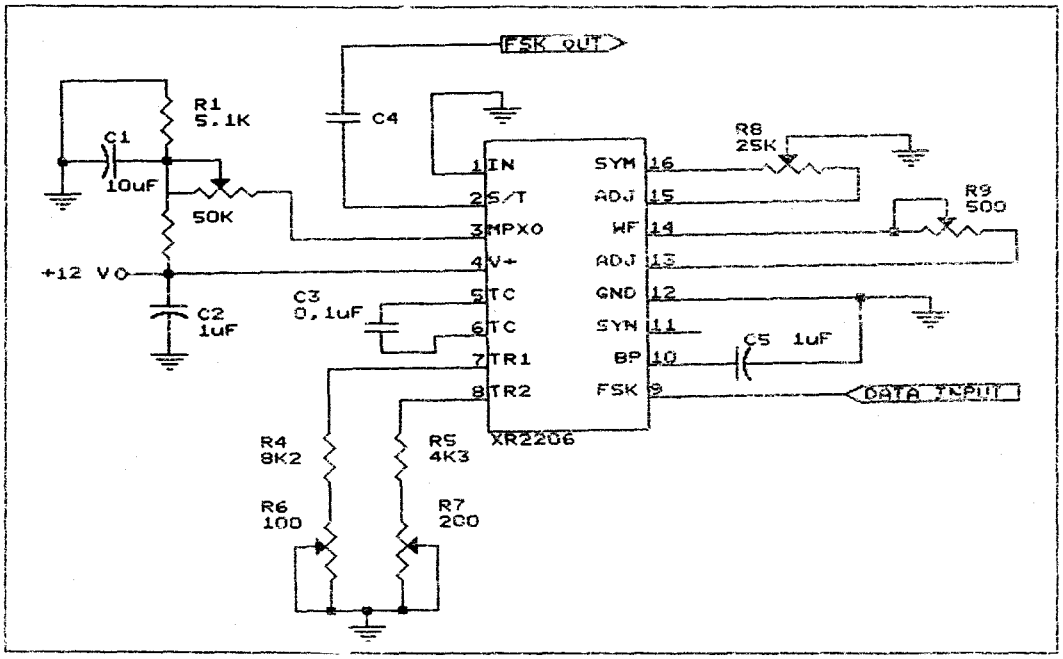
Selain memenuhi persyaratan di atas modem harus pula ditinjau dari sudut ekonomi :

- Peralatan modem harus mudah didapat, sehingga mudah untuk memasyarakatkannya.
- Harganya relatif murah.
- Rangkaian dibuat sesedarhana mungkin sehingga memudahkan untuk diperbanyak secara massal.

Berdasarkan semua kriteria di atas, maka digunakan komponen XR-2206 untuk modulator dan XR-2211 sebagai demodulator. Dalam perencanaan modul sistem ini, selanjutnya akan dibahas mengenai XR-2206 dan XR-2211.

3.1.3 XR 2206 FSK MODULATOR

Transmisi data melalui udara dari satu DTE (*Data Terminal Equipment*) yang satu ke DTE yang lain menggunakan metode *Frequency Shift Keying* (FSK). Di sini direncanakan frekuensi sebesar 2200 Hz untuk mewakili logika '0' dan 1200 Hz untuk mewakili logika '1'.



GAMBAR 3-2

SINUSOIDAL FSK GENERATOR

XR-2206 dioperasikan untuk menghasilkan dua frekuensi yang berbeda dengan menggunakan R_1 dan R_2, R_1 digunakan untuk menentukan frekuensi yang mewakili keadaan 'mark'; sedangkan R_2 dipakai untuk menentukan frekuensi yang mewakili keadaan 'space'. Pengaturan frekuensi untuk

'mark' dan 'space' dapat dilakukan dengan mengubah nilai dari R_1 dan R_2 pada pin 7 dan 8 tanpa saling tergantung satu sama lain. Frekuensi yang dihasilkan dapat diatur dengan menggunakan rumus sebagai berikut :[55)]

$$f_1 = \frac{1}{R_1 \; C} \text{ Hz}$$

$$f_2 = \frac{1}{R_2 \; C} \text{ Hz}$$

Sedangkan pengaturan amplitudo sinyal analog dapat dilakukan dengan mengubah-ubah nilai dari R_3.

Sinyal digital '1' dan 'Ø' yang akan dikirimkan diinputkan ke pin 9, jika pin 9 ini terbuka atau dihubungkan dengan tegangan $\geq$ 2 volt, maka hanya R_1 yang aktif; dan bila pin 9 diberi tegangan $\leq$ Ø,8 volt, maka R_2 yang aktif.

Langkah-langkah pada perencanaan sistem ini adalah sebagai berikut :

1. Mula-mula dipilih f_1 sebesar 1200 Hz untuk mewakili keadaan 'mark' dengan menggunakan rumus :

   $$f_1 = \frac{1}{R_1 \; C} \text{ Hz}$$

   dipilih nilai C sebesar Ø,1 μF; pada pin 5 dan 6 sehingga didapatkan nilai R_1 sebesar :

   $$R_1 = \frac{1}{1200 \times Ø,0000001} = 8333,33 \; \Omega$$

   maka R_1 diperoleh dengan jalan merangkai seri tahanan sebesar 8K2 Ω dengan tahanan variabel sebesar

---

55) Tomasi, W., **FUNDAMENTALS OF ELECTRONIC COMMUNICATIONS SYSTEMS**, Prentice Hall, New Jersey, 1988, hal. 299

100 Ω. Selanjutnya dengan mengatur tahanan variabel tersebut akan diperoleh frekuensi f_1 seperti yang diinginkan.

2. Kemudian dipilih f_2 sebesar 2200 Hz untuk mewakili keadaan 'space', digunakan rumus :

$$f_2 = \frac{1}{R_2 \; C} \text{ Hz}$$

seperti telah diketahui pada langkah 1, bahwa nilai C sebesar 0,1 μF; maka R_2 didapatkan sebesar :

$$R_2 = \frac{1}{2200 \times 0{,}0000001} = 4545{,}45 \; \Omega$$

maka R_2 dapat dihasilkan dengan jalan merangkai seri tahanan sebesar 4K3 Ω dengan tahanan variabel sebesar 200 Ω.

3. Untuk keperluan pengaturan bentuk gelombang, maka pada pin 13 dan 14 dipasang tahanan variabel sebesar 500 Ω.
4. Sedangkan untuk pengaturan kesimetrian gelombang, dipasang tahanan variable sebesar 25 KΩ pada pin 15 dan 16.
5. Pada pin 3 diberi rangkaian seperti pada gambar 3-2 yang berfungsi untuk keperluan multipleks output.
6. Pin 1 dihubungkan dengan ground karena tidak digunakan pada perencanaan ini.
7. Pin 4 merupakan Vcc oleh karenanya diberi filter C_2 sebagai penghilang ripple sebesar 1 μF. Sedangkan pin 12 merupakan ground power.

8. Pin 10 berfungsi sebagai bypass, oleh karenanya dihubungkan dengan ground setelah melalui kapasitor C_5 sebesar 1 μF.
9. Data digital diinputkan melalui pin 9, sedangkan output analog diperoleh dari pin 2 setelah melalui kapasitor kopling C_4 sebesar 0,047 μF.

3.1.4 XR 2211 FSK DEMODULATOR

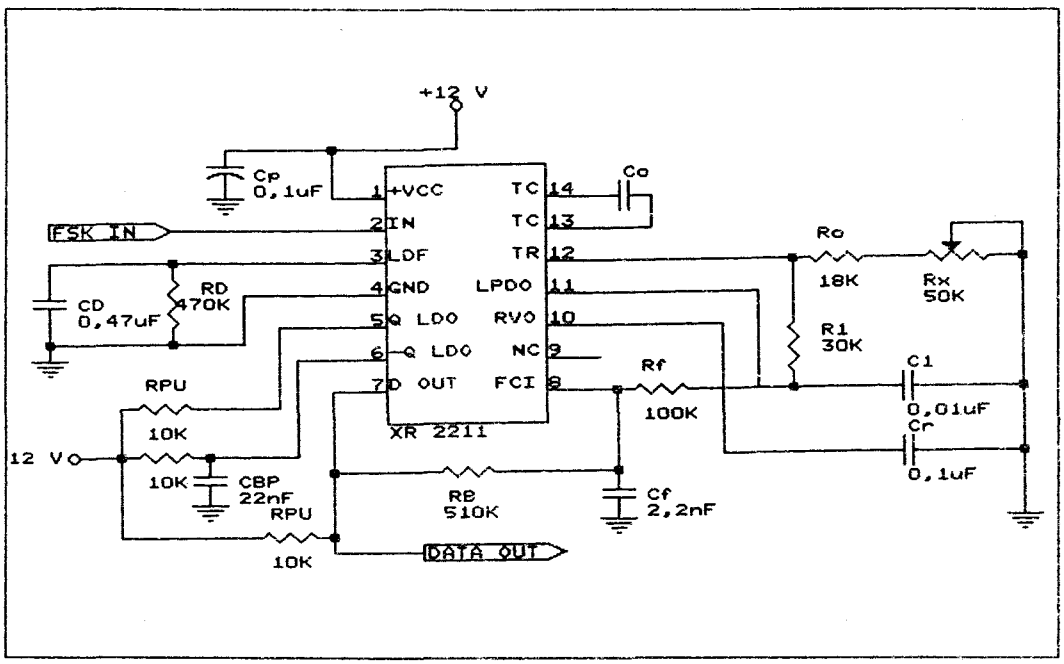
Peralatan ini berfungsi untuk mengkonversikan sinyal-sinyal analog yang masuk pada pin inputnya (pin 2) menjadi sinyal digital dengan cara membandingkan frekuensi dari sinyal analog tersebut dengan frekuensi tengah yang telah ditentukan. Penentuan frekuensi tengah tersebut dapat dilakukan dengan menggunakan rumus sebagai berikut :[50]

$$fo = \frac{1}{Ro\ Co}\ Hz$$

dimana kapasitor dalam Farad dan tahanan dalam Ohm.
Pada perencanaan ini fasilitas *Carrier Detect* (CD) tidak digunakan (pin 5 dan pin 6), sehingga *tracking* frekuensi tidak terlalu kritis. Setiap frekuensi di atas fo akan dianggap 'space' sedangkan jika frekuensi di bawah fo akan dianggap 'mark' tanpa memperhatikan adanya sinyal *carrier*.
Pin 3 dan pin 4 digunakan sebagai filter untuk

---

[50] Ibid, hal. 193



GAMBAR 3-3

FSK DEMODULATOR

menghilangkan *chatter* yang tidak diinginkan.

Pin 7 digunakan sebagai output data digital.

Adapun langkah-langkah perancangan untuk frekuensi 1200 Hz dan 2200 Hz adalah sebagai berikut :

1. Mula-mula ditentukan fo, yaitu :

   $$fo = (1200 + 2200) \times 1/2 = 1700 \text{ Hz}$$

2. Kemudian dipilih Co sebesar 27 nF, dan dipasang pada pin 13 dan 14

3. Selanjutnya dicari Ro dengan menggunakan rumus :

   $$Ro = \frac{1}{Co\ fo} = \frac{1}{(0{,}000000027 \times 1700)} = 21786\ \Omega$$

   Ro diperoleh dengan merangkai seri tahanan sebesar 18

KΩ dengan tahanan variabel sebesar 5Ø KΩ (RX) dan dipasang pada pin 12 dan ground. Menurut data book Ro mempunyai toleransi nilai antara 1Ø KΩ sampai 1ØØ KΩ. Ro dan Co lebih tepatnya dapat disebutkan sebagai pengontrol input VCO.

4. R_1 digunakan untuk membuat Δf sama dengan deviasi mark dan space, R_1 dicari dengan menggunakan rumus :[57)]

   $\Delta f/fo = Ro/R_1$

   $R_1 = Ro[fo/(f_1-fz)]$

   $R_1 = 18ØØØ\ [17ØØ/1ØØØ] = 3Ø6ØØ\ \Omega$

   Dengan menyesuaikan komponen yang ada di pasaran, maka dipilih R_1 sebesar = 3Ø KΩ.

   R_1 dipasang antara pin 12 dan pin 11.

5. Kemudian dicari nilai C_1 yang berfungsi untuk mengatur loop peredaman (*damping*). C_1 ditentukan dengan menggunakan rumus :

   $$C_1 = \frac{Co}{16\zeta/2} = Ø{,}25\ Co\ ,$$

   dimana ζ = damping ratio = 1/2

   sehingga diperoleh C_1 = 1Ø nF.

6. Selanjutnya ditentukan nilai R_D dan C_D, nilai R_D dipilih sebesar 47Ø KΩ. Sehingga C_D dapat dicari dengan memakai rumus :

   $C_D\ (\mu F) \geq (16/\text{range frekuensi dalam Hz})$

   karena range frekuensi adalah 1ØØØ Hz (22ØØ -12ØØ),

[57)] Ibid, hal. 191

didapatkan :

$C_D \geq (16/1000)$

$C_D \geq 0{,}016\ \mu F$

maka dipilih :

$C_D = 0{,}47\ \mu F$.

7. Langkah berikutnya adalah mencari nilai R_F, R_B, dan C_F. Dipilih nilai R_F = 100 KΩ dan R_B = 510 KΩ, sehingga C_F dapat dicari dengan menggunakan rumus :

   $C_F = 3/(\text{Baud Rate})\ ,\ \mu F$

   direncanakan baud rate-nya sebesar 1200 bps, maka :

   $C_F = 3/1200 = 0{,}0025\ \mu F$

   Dengan menyesuaikan nilai komponen yang ada di pasaran, maka dipilih C_F sebesar = 0,0022 μF.

8. Pin 10 dipakai sebagai tegangan referensi internal (V_R) terhadap pin 5, 8, 11, dan 12. Dimana V_R sebesar $V_R = V+/2 - 650$ mV. Pin ini di-bypass ke ground dengan menggunakan kapasitor sebesar 0,1 μF.

9. Pin 5 dan 6 (*Carrier Detect*) karena tidak dipakai, maka di-pull up ke Vcc melalui resistor 10 KΩ. Khusus untuk pin 6 selain di-pull up, juga di-bypass ke ground dengan menggunakan kapasitor sebesar 22 nF.

10. Pin 11 merupakan output dari *Phase Detector* dengan impedansi tinggi, dimana jika tidak ada sinyal atau tidak ada phase error, maka outputnya akan sebesar V_R. Tegangan outputnya berkisar antara $+V_R$ dan $-V_R$.

3.2 PERENCANAAN HUBUNGAN RS-232-C

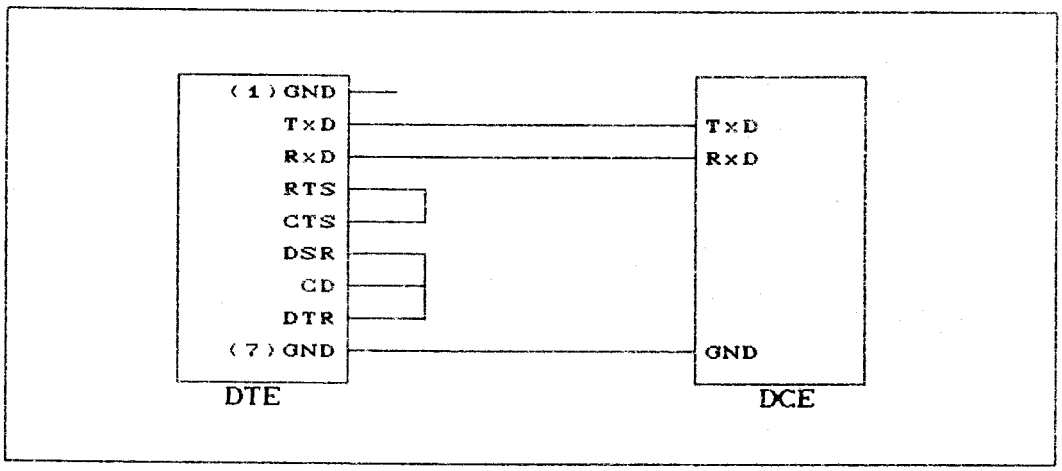
Sebelum mulai merencanakan hubungan RS-232-C harus diperhatikan terlebih dahulu beberapa kekurangan yang terdapat pada RS-232-C, adapun kekurangan tersebut adalah :

1. Panjang kabel penghubung antar DTE (*Data Terminal Equipment*) dibatasi oleh kapasitansi stray, yang mana dalam hal ini kapasitas stray maksimum adalah 2500 pF. Jadi bila kabel memiliki kapasitansi 50 pF/foot, maka panjang kabel maksimum adalah 50 feet (15 meter).
2. RS-232-C memiliki kecepatan pengiriman data maksimum sebesar 20000 bps.
3. Bila level sinyal ground antara kedua komputer / DTE yang letaknya berjauhan tidak sama, maka sinyal data akan terletak pada daerah transisi.

Setelah memperhatikan beberapa kekurangan di atas, maka direncanakan suatu hubungan RS-232-C yang mempergunakan kabel dengan panjang kurang dari 15 meter dan direncanakan kecepatan pengiriman data yang dipakai sebesar 300 bps.

Pada perencanaan ini digunakan hubungan RS-232-C tanpa handshaking. Pengaturan hubungan ini dilakukan pada pin - pin dari konektor DB-25. Pin 4 (RTS) dihubungkan dengan pin 5 (CTS), sehingga CTS akan aktif

begitu RTS diaktifkan. Pin 6 (DSR) dengan pin 8 (DCD), dan pin 20 (DTR) dihubungkan bersama agar pada saat IBM PC diaktifkan, maka output DTR, input DSR dan input DCD akan aktif pula. Sedangkan pin 7 (GND) dihubungkan dengan ground dari modem. Pin 2 (TxD) dihubungkan dengan bagian pengirim pada modem, dan pin 3 (RxD) dihubungkan dengan bagian penerima pada modem. Pin 1 (Protective Case Ground) tidak harus dihubungkan dengan ground case dari modem, pada perencanaan ini pin tersebut tidak digunakan. Gambar 3-4 menunjukkan hubungan tersebut.



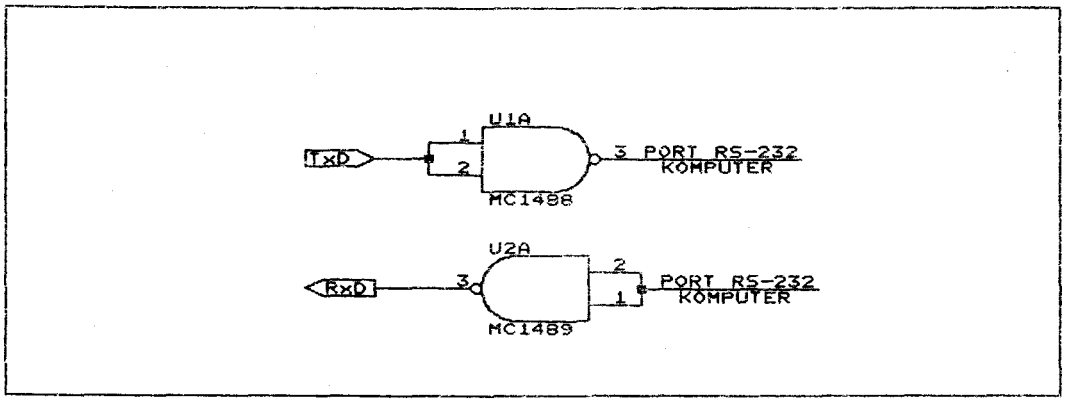
GAMBAR 3-4

HUBUNGAN RS-232-C TANPA HANDSHAKING

3.2.1 RANGKAIAN PENGUBAH LEVEL TEGANGAN RS-232-C

Sebelum data serial diterima oleh CPU atau peripheralnya, data tersebut harus diubah terlebih dahulu ke dalam standar level TTL dan sebaliknya sebelum

data dari CPU ditransmisikan, harus diubah terlebih dahulu ke level RS-232-C. Untuk mengubah level RS-232-C (pada kabel/saluran transmisi) menjadi standar TTL (pada CPU dan peripheralnya), dalam perencanaan ini digunakan IC MC 1489 dan untuk mengubah level TTL ke level RS-232-C digunakan MC 1488. Level TTL logika '0' didefinisikan pada 0 - 0,4 V, logika '1' didefinisikan pada level 2,4 - 5 V. Pada standar level RS-232-C logika '0' didefinisikan pada tegangan +3 V - +15 V, sedang logika '1' didefinisikan pada level (-3 V) - (-15 V). Rangkaian pengubah level tersebut dapat dilihat pada gambar 3-5.



GAMBAR 3-5

RANGKAIAN PENGUBAH LEVEL

3.3 PERENCANAAN PEMANCAR FM-STEREO

Perencanaan sistem pemancar radio FM - Stereo yang akan dipakai adalah menggunakan rangkaian yang

pernah ada di pasaran beberapa waktu yang lalu dengan mengadakan sedikit modifikasi pada bagian pembangkit frekuensi 19 kHz dan 38 kHz. Pada sistem ini yang harus diperhatikan adalah adanya perbedaan bila dibandingkan dengan pemancar pada band lain. Perbedaan itu adalah bahwa untuk memancarkan sinyal-sinyal pada band FM, maka gelombang pembawanya harus berada di sekitar 87,5 MHz dan 1Ø8 MHz. Untuk selanjutnya frekuensi tersebut dimodulasi. Pemodulasian frekuensi pembawa ini sering dikenal sebagai sinyal MPX (stereo multiplex). Ada tiga permasalahan pokok di sini yakni :

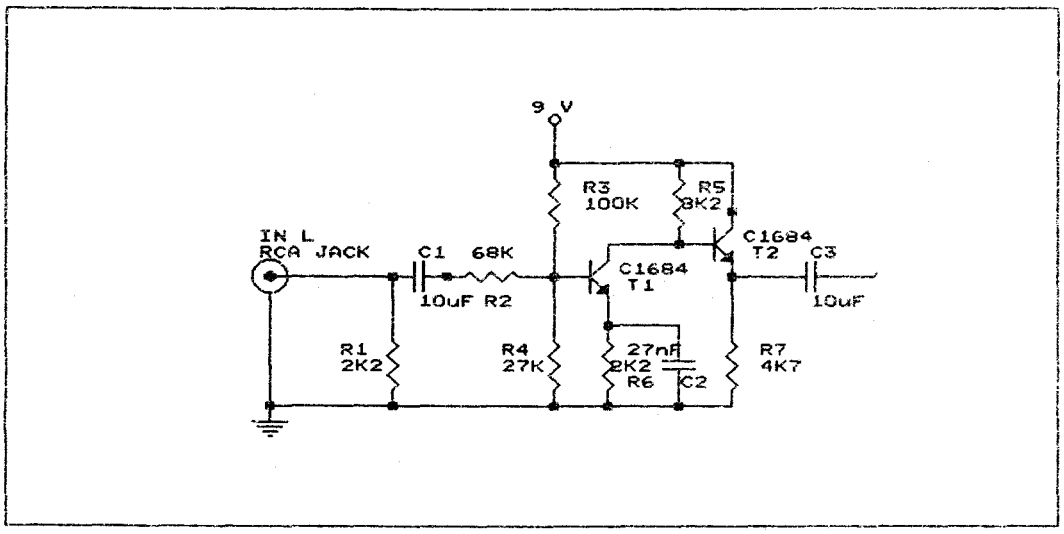
- penjumlahan sinyal (L+R)
- suatu perbedaan sinyal (L-R) yang dimodulasi pada 38 kHz
- stereo pilot tone 19 kHz

Penjumlahan sinyal yang dimaksudkan di sini adalah penjumlahan sinyal yang berada pada kanal kanan dan kiri (telah berada dalam bentuk sinyal audio). Lebar band dari sinyal-sinyal tersebut dipotong pada frekuensi antara sekitar 15 Hz sampai dengan 3Ø kHz. Untuk sistem penerima tunggal (mono), maka sinyal hanya terdapat pada satu sisi band saja. Sedangkan untuk informasi stereo, maka sinyal-sinyal yang ditransmisikan (setelah amplitudenya dimodulasi) akan berbeda antara sinyal yang berada dalam kanal L (*Left*, kiri) dan R (*Right*, kanan).

sinyal pada kanal L dan R berada dalam dua sisi band, yaitu pada 23 - 39,9 kHz dan 38,Ø3 - 53 kHz. Frekuensi 38 kHz tidak dipancarkan dengan maksud untuk mengefisienkan pemancar.

3.3.1 RANGKAIAN MULTIPLEX

Respon frekuensi untuk sinyal radio berada di atas 3 kHz, berasal dari pra penguat emphasis (pre-emphasis). Pengaruh penguatan pada frekuensi tinggi dari masukan emphasis akan membantu untuk mendapatkan

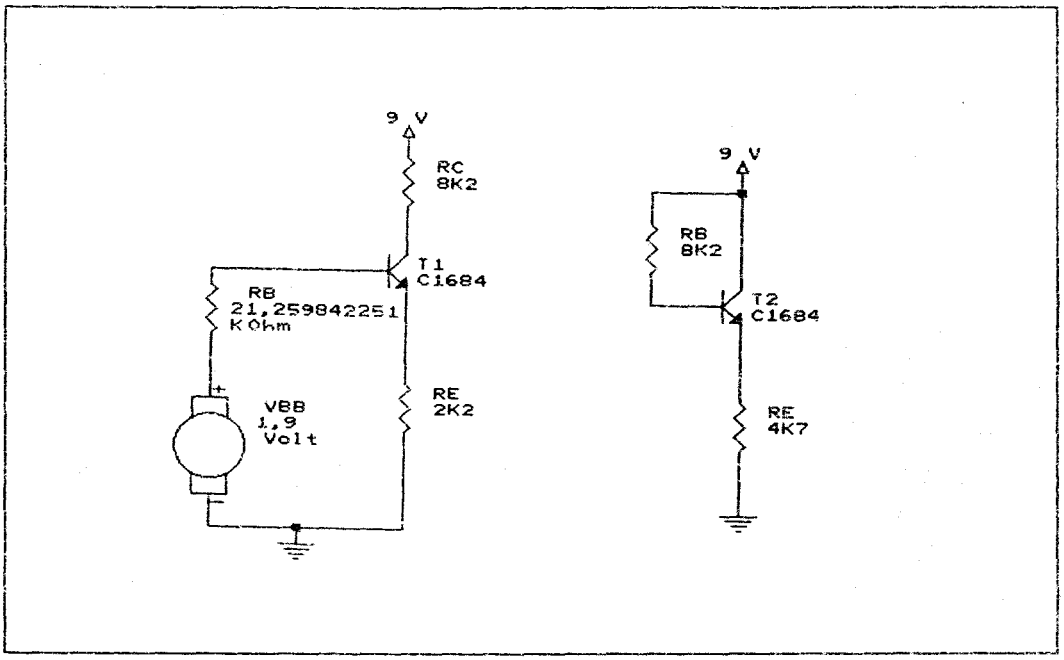


GAMBAR 3-6

RANGKAIAN PRE-EMPHASIS

perbandingan antara sinyal dan noise yang ideal. Pada bagian penerima, jaringan deemphasis mempunyai karakteristik dengan respon frekuensi hampir pada semua

bidang. Rangkaian yang akan dipakai pada perencanaan ini menitikberatkan pada penekanan *subcarrier*. Pada gambar 3-6 tampak bahwa sinyal-sinyal audio yang berada pada kanal L dan R, mula-mula dimasukkan pada jaringan *pre-emphasis*. Pada jaringan ini sinyal-sinyal audio ini diperkuat pada frekuensi menengah. Adapun analisa perencanaannya adalah sebagai berikut :



GAMBAR 3-7

RANGKAIAN PENGGANTI THEVENIN UNTUK RANGKAIAN PREEMPHASIS TRANSISTOR T1 DAN T2

Mula-mula direncanakan terlebih dahulu titik kerja dari transistor T1 dan T2 pada daerah aktif, sedangkan penguatan tegangan input total dari T1 dan T2 adalah

sebesar 15 sampai dengan 20 kali. Untuk memenuhi kriteria tersebut maka transistor T1 dan T2 dirangkai dengan konfigurasi Kaskade *Common Emitor* (CE) dan *Common Collector* (CC) seperti terlihat pada gambar 3-7. Dari rangkaian *Common Emitor* dipilih komponen - komponen pendukung sehingga titik kerjanya berada pada daerah aktif :

$$V_{TH} = \frac{27}{127} \times 9 \text{ Volt} = 1{,}9134 \text{ Volt}$$

$$R_{TH} = 27\ K\Omega \,//\, 100\ K\Omega = 21{,}25984251\ K\Omega$$

Selanjutnya dilihat loop input pada rangkaian pengganti transistor T_1 :

$$V_{BB} = I_B.R_B + V_{BE} + I_E.R_E$$

$$= I_B R_B + V_{BE} + (1 + \beta dc)\, I_B R_E$$

dimana $\beta dc = \beta ac = 100$, maka

$$1{,}9 = (I_B \times 21{,}25984 \times 10^3) + 0{,}6 + (101 \times I_B \times 2200)$$

$$1{,}3 = (222200 + 21259{,}84)\, I_B$$

$$I_B = \frac{1{,}3}{(243459{,}84)} = 5{,}339\ \mu A$$

Sehingga diperoleh :

$$I_C = \beta dc\, I_B = 100 \times 5{,}339\ \mu A = 533{,}9\ \mu A$$

$$I_E = (1 + \beta dc)\, I_B = 101 \times 533{,}9\ \mu A = 539{,}239\ \mu A$$

Langkah berikutnya adalah menentukan tegangan V_{CE} dengan melihat loop output :

$$V_{CC} = I_C.R_C + V_{CE} + I_E.R_E$$

$$= \beta I_B.R_C + V_{CE} + (1 + \beta dc)\, I_B.R_E$$

$= (533,9 \times 10^{-6} \times 8200) + V_{CE} + (539,239 \times 10^{-6} \times 2200)$

$V_{CE} = 3,4357$ Volt

Sehingga titik kerja dari rangkaian CE adalah :

(3,4357 V, 0,5339 mA)

Untuk rangkaian *Common Collector* dipilih juga komponen - komponen sehingga titik kerjanya berada pada daerah aktif , mula - mula dilihat pada loop input :

$V_{CC} = V_{RB} + V_{BE} + V_{RE}$

$= I_B.R_B + V_{BE} + I_E.R_E$

$= I_B.R_B + V_{BE} + (1 + \beta dc)\ I_B.R_E$

$= (I_B \times 8200) + 0,6 + ((1 + 100)\ I_B\ 4700)$

$= 482900\ I_B$

$I_B = 17,39491\ \mu A$

$I_C = \beta dc\ I_B = 100 \times 17,39491\ \mu A = 1739,491\ \mu A$

$I_E = (1 + \beta dc)\ I_B = 101 \times 17,39491\ \mu A = 1756,886\ \mu A$

kemudian pada loop output :

$V_{CC} = V_{CE} + V_{RE}$

$= V_{CE} + I_E.R_E$

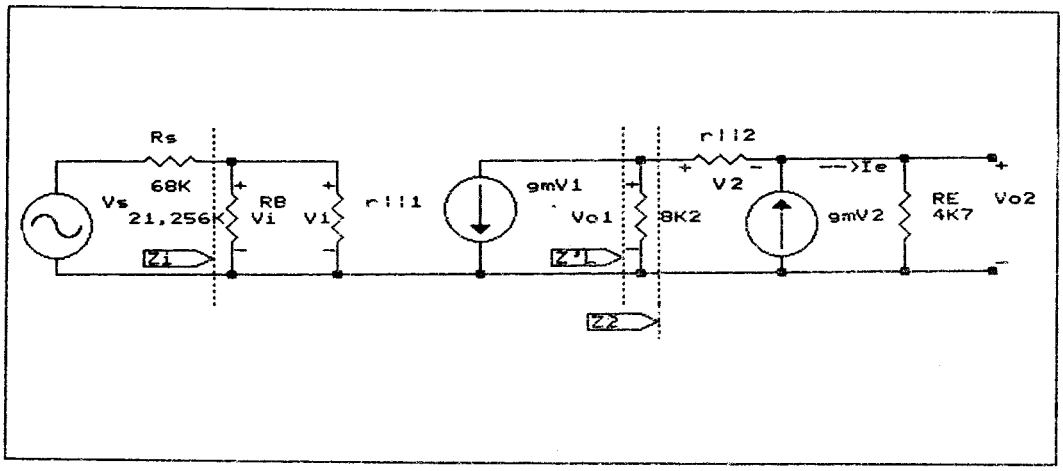
$= V_{CE} + (1756,886 \times 10^{-6} \times 4700)$

$V_{CE} = 0,743$ Volt

Titik kerjanya adalah : (0,743 V, 1739,491 μA).

Kemudian dilanjutkan dengan perhitungan penguatan dari konfigurasi transistor tersebut. Konfigurasi

kaskade *Common Emitor* dan *Common Collector* mempunyai rangkaian ekivalen seperti pada gambar 3-8.



GAMBAR 3-8

RANGKAIAN EKIVALEN FREKUENSI MENENGAH

KONFIGURASI KASKADE COMMON EMITOR DAN COMMON COLLECTOR

Dari data book ditetapkan $\beta ac = \beta o = 100$, perhitungan penguatan dari rangkaian di atas adalah sebagai berikut:

mula - mula didapat gm sebesar = $38,9.|Ic|$

$= 38,9 \times 533,9 \times 10^{-6}$

$= 20,77$ mmho.

kemudian $r\pi = \frac{\beta o}{gm} = \frac{100}{20,77 \times 10^{-3}} = 4,815$ KΩ

$Ie = \frac{Vz}{r\pi}(1 + \beta o)$

$Voz = 4700 \times Ie = 4700 \times \left(\frac{Vz}{r\pi}(1 + \beta o) \right)$

$Voz = \frac{474700}{4,815} \times 10^{-3} \times Vz = 9,8588.Vz.....(1)$

$Vo1 = Vz + Voz = Vz + \frac{4700}{1000} \times \left(\frac{101}{4,815}\right) Vz$

$V_{o1} = 99,5878\ V_2$ .....(2)

$$Z_2 = \frac{V_2 + V_{o2}}{\frac{V_2}{r\pi}} = \frac{99,5878\ V_2}{\frac{V_2}{4815}} = 479,515257\ K\Omega.$$

$Z'_L = R_{L1} // Z_2 = 8,Ø6213\ K\Omega$

$V_{o1} = g_m V_1.Z'_L = 2Ø,77 \times 1Ø^{-3} \times 8,Ø6213 \times 1Ø^{3} \times V_1$

$V_{o1} = 167,45Ø4\ V_1$.....(3)

dimana, $V_1 = \frac{Z_i}{R_s+Z_i} \times V_s$

sedangkan $Z_i = R_B // r\pi = 3,9259\ K\Omega$

sehingga $V_1 = \frac{3,9259\ K\Omega}{(68 + 3,9259)K\Omega} \times V_s = Ø,Ø5458\ Vs$...(4)

Selanjutnya persamaan (4) disubstitusikan ke persamaan (1), diperoleh : $V_{o1} = 9,13944V_s$.....(*), persamaan (*) ini disubstitusikan ke (2) akan dihasilkan :

$9,13944\ V_s = 99,5878\ V_2$

$V_2 = \frac{9,13944}{99,5878} \times V_s = Ø,Ø9177\ V_s$.....(**)

Untuk selanjutnya persamaan (**) ini disubstitusikan ke persamaan (1), hasilnya :

$V_{o2} = 9,8588 \times Ø,Ø9177\ V_s = Ø,9Ø4742\ V_s$

Sehingga $A_{vs} = \frac{V_{o2}}{V_s} = \underline{Ø,9Ø4742}$ kali.

maka $A_{vi} = \frac{R_s+Z_i}{Z_i} . A_{vs} = 18,321 \times Ø,94742 = 17,35768$

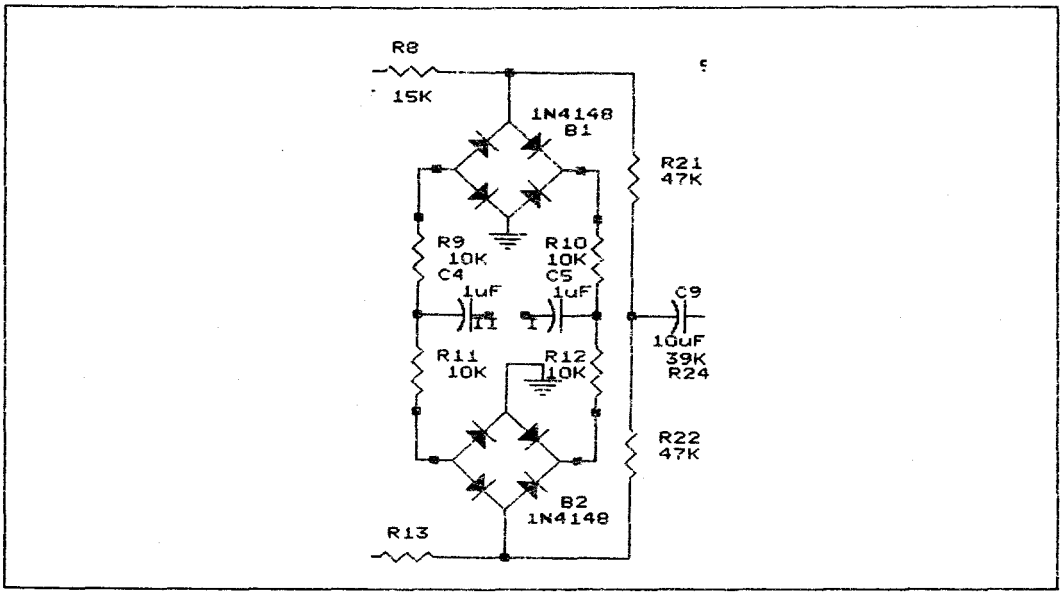
$A_{vi} = \frac{V_{o2}}{V_i} = \underline{17,35768}$ kali

$A_I = A_{vi} \frac{Z_i}{R_L} = 17,35768 \times \frac{3925,9}{47ØØ} = \underline{14,49883}$ x

$A_P = A_{vi} \times A_I = 17,35768 \times 14,49883 = \underline{251,67}$ x

Selanjutnya sinyal-sinyal tersebut dimasukkan pada rangkaian pengendali (*chopper modulator*) dengan frekuensi 38 kHz. Perlu diketahui bahwa frekuensi

pengendalian tersebut diperoleh dengan mengalikan dua output dari oscillator 19 kHz. Sinyal-sinyal L dan R pada kenyataannya merupakan setengah siklus dari frekuensi 38 kHz. Penjumlahan sinyal yang berada pada kanal L dan R menghasilkan sinyal MPX. Sinyal ini hanya akan berada pada satu sisi band saja. Gambar 3-9 menunjukkan rangkaian pengendali tersebut. Frekuensi 38 kHz yang dibangkitkan oleh IC 4011 mempunyai bentuk gelombang kotak (*square wave*). Sinyal L dan R akan dilewatkan oleh rangkaian pengendali pada setiap



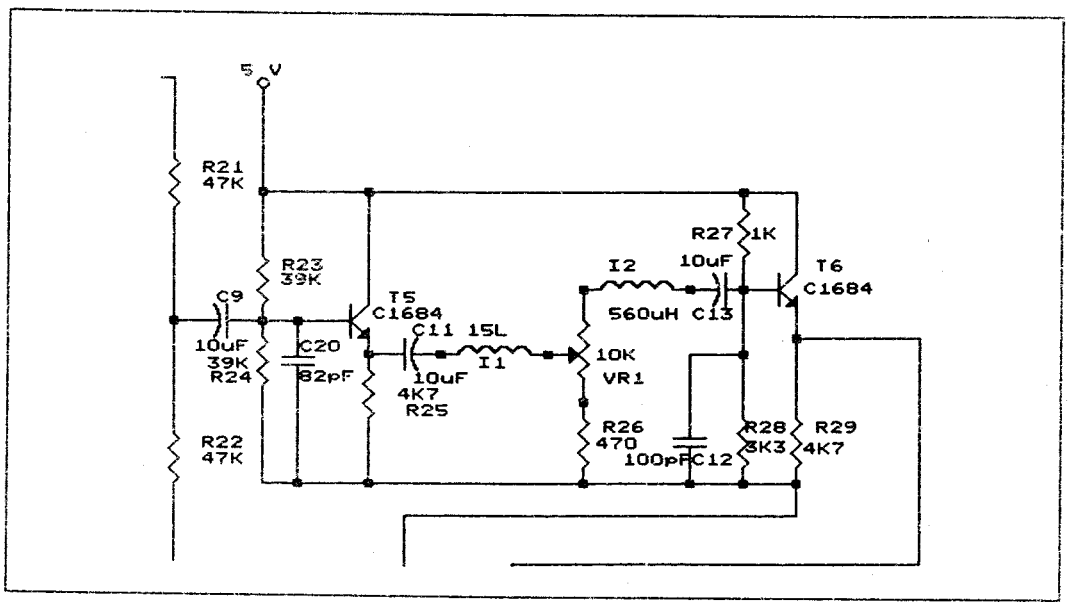
GAMBAR 3-9

RANGKAIAN CHOPPER MODULATOR

setengah siklus dari frekuensi 38 kHz, jadi secara bergantian sinyal L dan R dilewatkan oleh rangkaian

pengendali (*Chopper Modulator*) menuju ke rangkaian pencampur. Sinyal MPX yang dihasilkan ini hanya berada pada satu sisi band saja.

Berikut ini akan dibahas rangkaian transistor yang berfungsi sebagai *Buffer* (penyangga). Rangkaian ini menggunakan konfigurasi *Common Collector*, dimana penguatan tegangan direncanakan sama dengan satu. Rangkaian tersebut dapat dilihat pada gambar 3-10.



GAMBAR 3-10

RANGKAIAN BUFFER

Titik kerja dari rangkaian buffer yang menggunakan transistor T5 adalah sebagai berikut :
Sumber daya Thevenin dari rangkaian input adalah sebesar 2,5 Volt, sedangkan resistansi Thevenin dari rangkaian

input adalah sebesar 19,5 KΩ.

Mula-mula dianalisa pada loop input :

$V_{BB} = I_B.R_B + V_{BE} + I_E.R_E$

$2,5 = I_B \times 19,5 \times 10^3 + 0,6 + (1+\beta dc) \times I_B \times 4700$

$1,9 = 494200\ I_B$

$I_B = 3,844597\ \mu A$

$Ic = \beta dc\ I_B = 100 \times 3,844597\ \mu A = 384,4597\ \mu A$

$I_E = (1 + \beta dc)\ I_B = 101 \times 3,844597\ \mu A = 388,3043\ \mu A$

sedangkan pada loop output :

$Vcc = V_{CE} + I_E.R_E$

$5 = V_{CE} + (388,3043 \times 10^{-6} \times 4700)$

$V_{CE} = 5 - 1,825 = 3,175$ Volt

Jadi titik kerjanya adalah : (3,175 V ; 384,4597μA)

Selanjutnya penguatan dari rangkaian tersebut diharapkan mendekati satu. Gambar 3-11 menjelaskan rangkaian pengganti ekivalen pada frekuensi menengah dari rangkaian tersebut. Sedangkan perhitungan dari komponen komponen yang dipilih adalah sebagai berikut :

Jika variable resistor VR_1 dalam keadaan maksimum berarti tahanan bebannya, RL sebesar 10470 Ω. Berarti

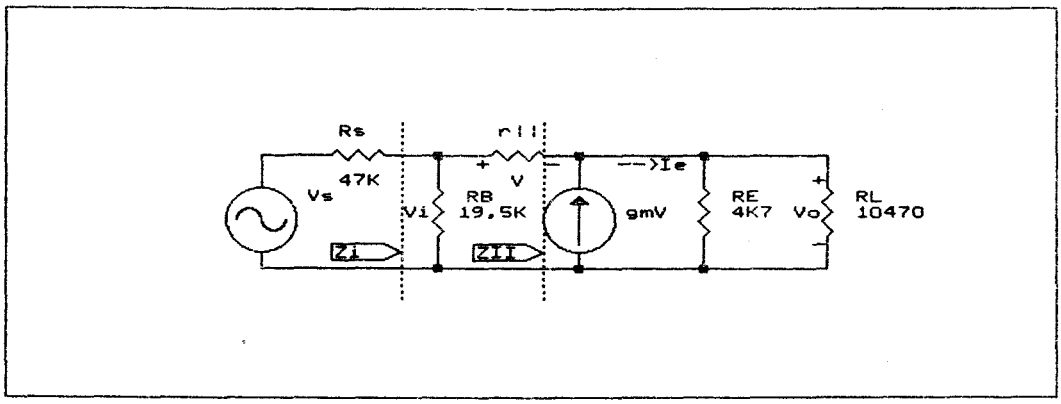
$R'L = R_E // R_L = 4700 // 10470 = 3,24384$ KΩ

sedang gm = 38,9 (Ic) = 38,9 x 384,4597 μA

= 14,956 mmho.

maka $r\pi = \frac{\beta o}{gm} = \frac{100}{14,956} = 6,68628$ KΩ

$Zi = R_B // (r\pi + Z_{II})$



GAMBAR 3-11

RANGKAIAN EKIVALEN FREKUENSI MENENGAH

KONFIGURASI COMMON COLLECTOR TRANSISTOR T5

$Z_{II} = (1 + \beta o)\ R'L = 101\ (3243,84) = 327,62784\ K\Omega$

$Z_i = 19,5\ K\Omega\ //\ 334,31412\ K\Omega = 18,42528\ K\Omega$

Selanjutnya dicari $Av_i = \frac{V_o}{V_i} = \frac{(1+\beta o).R'L}{r\pi + (1+\beta o).R'L}$

$$= \frac{327627,84}{334310,64} = \underline{0,98}\ x \simeq 1\ x$$

Jika $R_s = 47\ K\Omega$, maka $Av_s = Av_i . \frac{Z_i}{R_s + Z_i}$

$$Av_s = 0,98\ x\ \frac{18,42528\ K\Omega}{(47 + 18,42528)K\Omega} = \underline{0,27599}\ x$$

$$A_I = Av_i . \frac{Z_i}{R_L} = 0,98\ x\ \frac{18425,28}{10470} = \underline{1,7246}\ x$$

$$A_P = Av_i\ x\ A_I = 0,27599\ x\ 1,7246 = \underline{0,47579}\ x$$

Sedang bila variable resistor VR1 pada kedudukan minimum, maka tahanan bebannya (R_L) sebesar 470 Ω. Berarti $R'L = R_E\ //\ R_L = 4700//470 = 427,2727\ \Omega$.

$$Av_i = \frac{V_o}{V_i} = \frac{(1+\beta o).R'L}{r\pi + (1+\beta o).R'L} = \frac{43154,5427}{49840,8227}$$

$$= \underline{0,86585}\ x$$

$Z_i = R_B // (r\pi + Z_{II})$

$Z_{II} = (1 + \beta o)\ R'L = 43,1545427\ K\Omega$

$Z_i = 19,5\ K\Omega // 49,8408\ K\Omega = 14,016216\ K\Omega$

$R_s = 47\ K\Omega.$

maka $Av_s = Av_i \frac{Z_i}{R_s + Z_i} = \underline{0,198897}\ x$

$A_I = Av_i \frac{Z_i}{R_L} = \underline{25,8212}\ x$

$A_P = Av_i \times A_I = \underline{22,35729}\ x$

Selanjutnya sinyal MPX tersebut dilewatkan rangkaian filter induktor yang akan melewatkan sinyal dengan frekuensi di bawah 53 KHz, sehingga frekuensi - frekuensi harmonisa di atas 53 KHz tidak ikut dimodulasikan oleh oscillator.

Di antara rangkaian filter tersebut, terdapat rangkaian pembagi arus yang akan mengatur arus yang akan menuju ke rangkaian berikutnya. Pada saat variable resistor maksimum (10K) terhadap titik ground, maka arus yang menuju rangkaian berikutnya adalah maksimum. Dan terjadi sebaliknya apabila variable resistor minimum.

Selanjutnya sinyal MPX tersebut dibuffer lagi oleh rangkaian transistor T6 yang mempunyai rangkaian pengganti seperti pada gambar 3-12.

Transistor tersebut direncanakan mempunyai titik kerja sebagai berikut :

$V_{TH} = \frac{3K3\Omega}{4K3\Omega} \times 5\ Volt = 3,83721\ Volt$

$R_{TH} = 3K3\ //\ 1\ K = 767,442\ \Omega$

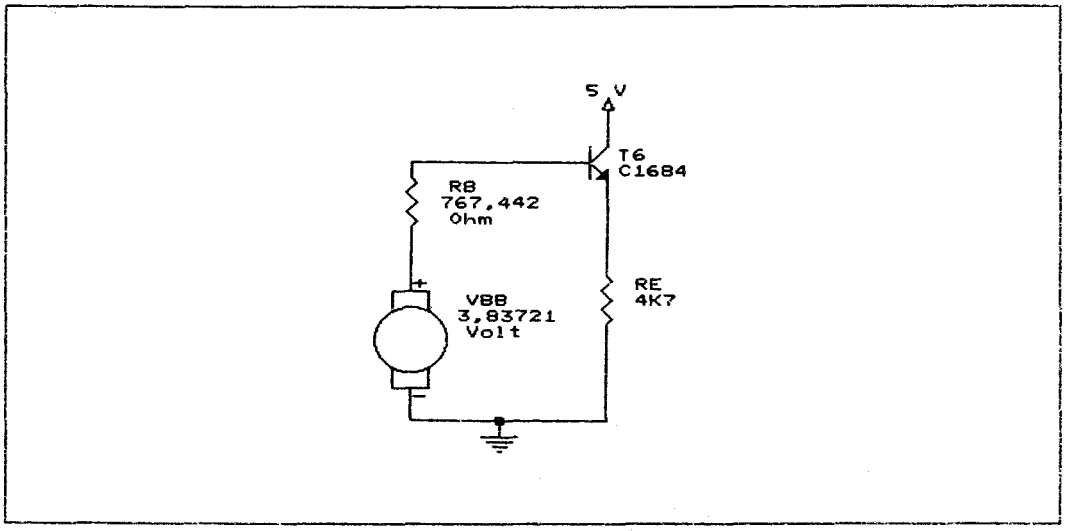
pada loop input :

$$V_{BB} = I_B.R_B + V_{BE} + I_E.R_E$$

$$= I_B.R_B + V_{BE} + (1 + \beta dc).R_E$$

dari data book diketahui $\beta dc = \beta o = 100$

$3,83721 = (I_B \times 767,442) + 0,6 + (101 \times I_B \times 4700)$



GAMBAR 3-12

RANGKAIAN PENGGANTI THEVENIN DARI

RANGKAIAN TRANSISTOR T6

$3,23721 = 475467,442\ I_B$

$I_B = 6,808479\ \mu A$

$I_C = \beta dc\ I_B = 100 \times 8,79332\ \mu A = 680,8479\ \mu A$

$I_E = (1 + \beta dc)\ I_B = 101 \times 8,79332\ \mu A = 687,656379\ \mu A$

selanjutnya pada loop output :

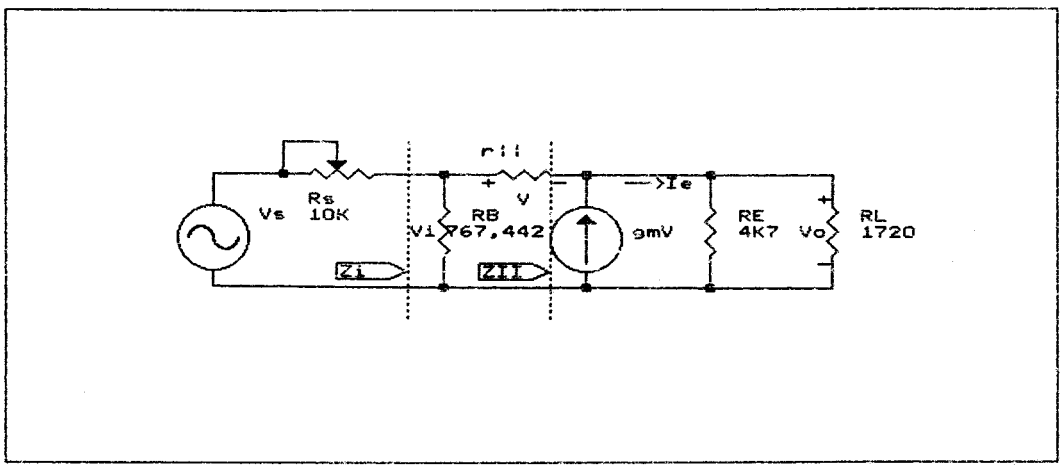
$V_{CC} = V_{CE} + I_E\ .\ R_E$

$5 = V_{CE} + (687,656379 \times 10^{-6} \times 4700)$

V_{CE} = 5 - 3,231985 = 1,768015 Volt

Maka titik kerjanya adalah (1,768015 V ; 687,656379 μA)

Selanjutnya dicari penguatan dari rangkaian buffer tersebut, tampak pada gambar 3-13 merupakan rangkaian ekivalen dari konfigurasi *Common Collector* transistor T6 tersebut dengan komponen - komponen yang dipilih agar mempunyai penguatan tegangan input sama dengan satu.



GAMBAR 3-13

RANGKAIAN EKIVALEN FREKUENSI MENENGAH

DARI KONFIGURASI COMMON COLLECTOR TRANSISTOR T6

Mula-mula diketahui dari data book, $\beta o = \beta dc =$ 100. Kemudian dicari gm, sebesar : 38,9 x Ic = 26,484983 mmho. Dan $r\pi = \frac{\beta o}{gm} = \frac{100000}{26,484983} = 3,775724$ KΩ.

$R'L$ = 4K7 Ω // 1720 Ω = 1259,19 Ω

Sehingga $Avi = \frac{Vo}{Vi} = \frac{(1+\beta o)RL}{r\pi + (1+\beta o)RL} = \frac{474700}{474702,9235}$

= 0,9921 x

kemudian dicari $Z_i = R_B // (r_\pi + Z_{II})$, $Z_{II} = (1+\beta_o)\, R'_L$

$$Z_i = 767{,}442\ \Omega\ //\ 127178{,}19\ \Omega$$

$$Z_i = 762{,}8387\ \Omega.$$

diperoleh $A_I = A_{vi} \dfrac{Z_i}{R_L} = \dfrac{756{,}8123}{1259{,}19} = \underline{0{,}601031}\ x$

$$A_P = A_{vi} \times A_I = \underline{0{,}59628}\ x$$

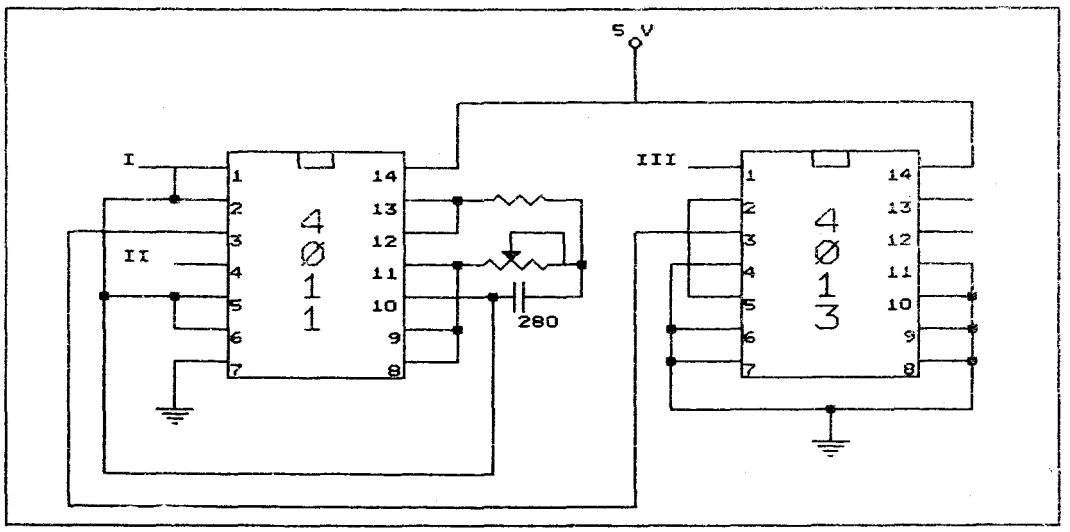
Bila Rs = VR = 10KΩ (maksimal), maka :

$$A_{vs} = A_{vi} \frac{Z_i}{R_s+Z_i} = 0{,}9921 \times \frac{762{,}8387}{10762{,}8387}$$

$$= \underline{0{,}070317}\ x$$

Bila Rs = VR = 0 (minimal), maka : $A_{vs} = A_{vi} = \underline{0{,}9921}\ x$

Sinyal stereo pilot tone 19 kHz dihasilkan oleh IC 4013 yang berfungsi sebagai pembagi 2 dari frekuensi 38 kHz yang dihasilkan oleh IC 4011. Sinyal ini selanjutnya akan dijumlahkan dengan sinyal MPX sebelum dimasukkan ke bagian oscillator FM. Gambar 3-14



GAMBAR 3-14

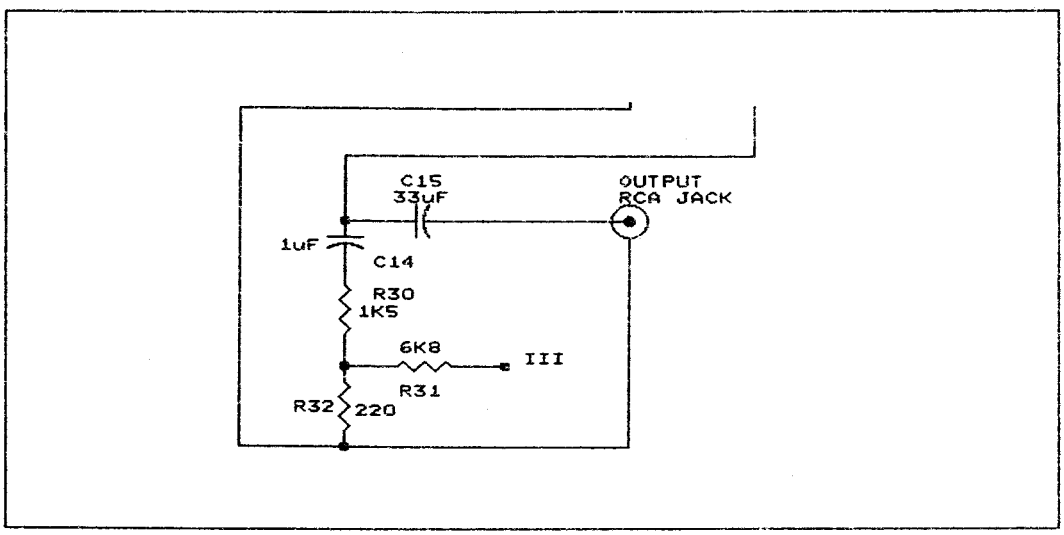
RANGKAIAN PEMBANGKIT FREKUENSI 19 KHz & 38 KHz

menunjukkan rangkaian penghasil *stereo pilot tone* dan frekuensi 38 kHz tersebut. Pada perencanaan sistem ini mula-mula dipilih kapasitor sebesar 280 pF, selanjutnya dihitung resistornya agar diperoleh frekuensi sebesar 38 kHz. Dari perhitungan didapatkan sebagai berikut : $R = \frac{1}{f \cdot C}$

$$= \frac{1}{38000 \times 28 \times 10^{-11}}$$

$$R = 93.985\ K\Omega$$

Sehingga dipilih Variable resistor sebesar 100 kΩ. Sementara itu untuk rangkaian pembangkit frekuensi 19 kHz dipakai rangkaian pembagi dua dengan menggunakan IC 4013 seperti tampak pada gambar 3-14.



GAMBAR 3-15

RANGKAIAN PENCAMPUR

Setelah melalui rangkaian buffer selanjutnya sinyal informasi akan dicampur dengan sinyal stereo pilot tone 19 kHz pada bagian pencampur seperti terlihat pada gambar 3-15. Selanjutnya output dari rangkaian pencampur ini dimasukkan ke bagian oscillator FM.

3.3.2 RANGKAIAN OSCILLATOR FM

Output dari rangkaian multiplex berupa sinyal MPX (multiplex), sinyal ini selanjutnya dimasukkan ke rangkaian oscillator FM. Oscillator FM tidak ubahnya suatu oscillator sederhana yang mempunyai denyutan frekuensi bervariasi antara 88 sampai dengan 108 MHz. Sinyal stereo MPX akan memodulasi sinyal oscillator ini untuk kemudian ditransmisikan melalui antena. Gambar 3-16 menunjukkan rangkaian oscillator FM yang dipakai. Oscillator yang dipakai pada sistem ini adalah oscillator Collpits dengan konfigurasi *Common Base* dan feed back pada emitornya. Lilitan L4 dan L5 berfungsi sebagai Transformer output, yang berfungsi untuk mengkonversikan sinyal menjadi pulsa-pulsa listrik. Selain itu rangkaian tanki dapat terbebani dengan ringan. Sedangkan untuk pemilihan komponen agar oscillator ini mempunyai range frekuensi antara 88 - 108 MHz maka dipakai cara sebagai berikut :

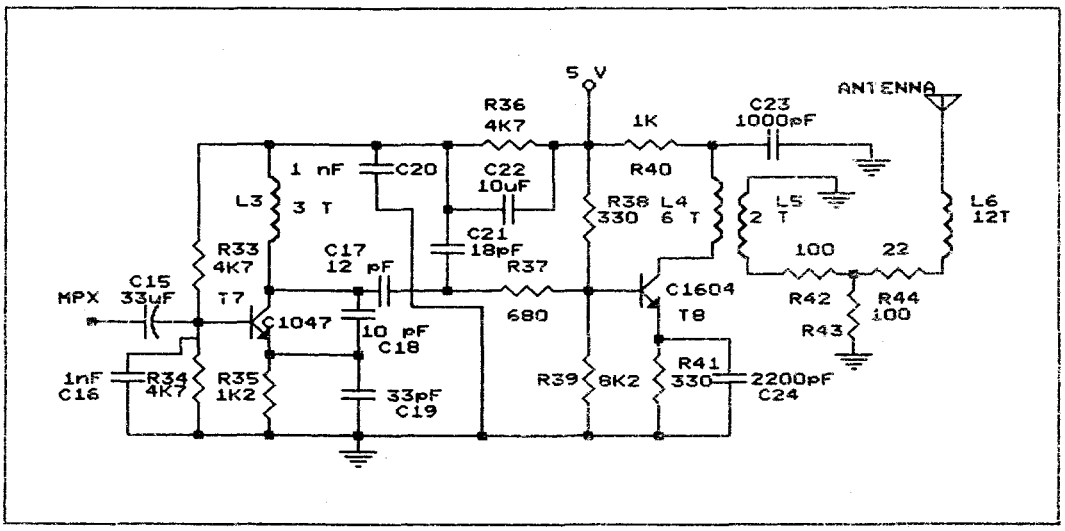
mula-mula dipilih kapasitor C14 = 10 pF dan C15 = 33 pF

hubungan seri kedua kapasitor ini menghasilkan C total sebesar : 7,67442 x 10^{-12} Farad. Sehingga untuk mendapatkan frekuensi sebesar 88 MHz, dicari induktor sebesar :

$$f = \frac{1}{2\pi\sqrt{L.C}}$$

$$88.10^{6} = \frac{1}{2\pi\sqrt{L \times 7,67442 \times 10^{-12}}}$$

$$L = 0,426\ \mu H$$



GAMBAR 3-16

RANGKAIAN OSCILLATOR

Sedangkan untuk f = 108 MHz , induktornya sebesar :

$$108.10^{6} = \frac{1}{2\pi\sqrt{L \times 7,67442 \times 10^{-12}}}$$

$$L = 0,28297\ \mu H$$

Sehingga untuk keperluan ini dipakai induktor dengan inti ferit (koker) sebanyak tiga lilitan agar mempunyai range nilai antara 0,28297 μH sampai dengan 0,426 μH.

3.4 PERENCANAAN PERANGKAT LUNAK

Perangkat lunak yang dipakai pada tugas akhir ini direncanakan untuk mengakses card adapter komunikasi serial pada komputer IBM PC, baik unit pengirim data maupun pada unit penerima data. Perangkat lunak ini berfungsi sebagai sarana utama untuk dapat bekerjanya sistem secara keseluruhan. Banyak bahasa pemrograman yang dapat dipergunakan untuk mendukung keperluan ini, antara lain : BASIC, TURBO PASCAL, TURBO C, ASSEMBLY dan lain-lain. Sebenarnya sudah ada paket perangkat lunak yang digunakan untuk komunikasi data, seperti misalnya : KERMIT dan PROCOMM. Akan tetapi pada perencanaan tugas akhir ini, perangkat lunak disusun dengan menggunakan bahasa ASSEMBLY. Perangkat lunak ini mempunyai kemampuan untuk mengakses alamat dan mengeksekusi register komunikasi pada IC INS 8250 yang terdapat pada card serial interface komputer IBM PC.

Program perangkat lunak ini menggunakan fasilitas interupt 0Ch untuk setiap data yang masuk. Adapun segment program untuk keperluan ini adalah sebagai berikut :

```
cli                                 ;mengutip vektor
mov al,0ch                          ;address int 0Ch asli
mov ah,35h
int 21h

mov ax,@data                        ;meletakkan kembali
mov ds,ax                           ;address int_service
mov si,offset adres_int_och         ;routine untuk mela -
```

```
mov word ptr ds:[si],bx      ;yani int ØCh.
mov word ptr ds:[si+2],es
mov dx,cs
mov ds,cx
lea dx,intr_servis
mov al,Och
mov ah,25h
int 21h
mov ax,@data
mov ds,ax
mov es,ax
```

Setelah dilakukan pengaturan vektor address untuk interrupt, selanjutnya dilakukan inisialisasi 825Ø untuk mengatur jumlah bit data, parity, baud rate dan lain-lain. Segment programnya adalah sebagai berikut :

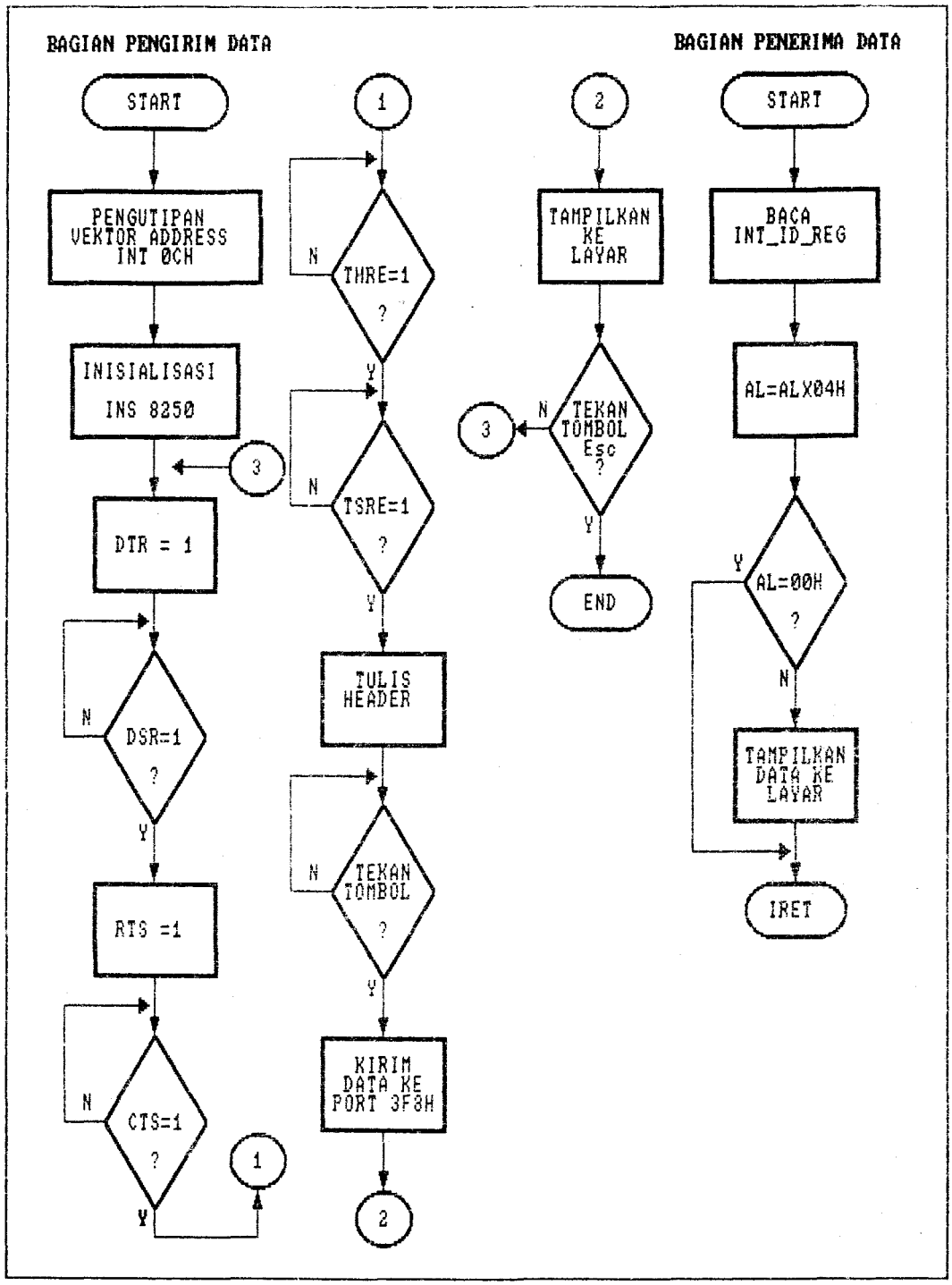
```
mov dx,LINE_CONT_REG    ;digunakan untuk mengak -
mov al,10000000b        ;ses BAUD_DIV_REG
out dx,al

mov dx,BAUD_DIV_MSB     ;baud rate diatur sebesar
mov al,01h              ;3ØØ bps
out dx,al

mov dx,BAUD_DIV_LSB
mov al,80h
out dx,al

mov dx,LINE_CONT_REG    ;1 frame terdiri dari 8
mov al,00000011b        ;bit data, parity disable,
out dx,al               ;1 stop bit, dan control
                        ;break tidak aktif.
mov dx,INT_ENBL_REG
mov al,01h
out dx,al
```

Gambar 3-17 menunjukkan flow chart dari program perangkat lunak yang digunakan pada sistem secara keseluruhan, sedangkan urutan program selengkapnya terdapat pada lampiran.



GAMBAR 3-17

FLOW CHART PROGRAM

BAB IV

PENGUJIAN ALAT DAN SISTEM KESELURUHAN

4.1 PENGUJIAN PROGRAM KOMUNIKASI

Pengujian program komunikasi dilakukan dengan menggunakan bahasa pemrograman ASSEMBLY, dimana dilakukan pengujian berulang-ulang dengan menggunakan media kabel, dengan cara mengubah-ubah baud rate mulai dari 50, 75, 110, 134,5 , 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600.

Format data dalam satu *frame* dipilih sebanyak 8 bit data ditambah 1 stop bit dan tidak menggunakan parity check. Pada baud rate 50, data - data yang dikirimkan cukup lambat sampai pada komputer penerima, kecepatan penerimaan data bertambah dengan bertambahnya baud rate. Data-data yang dikirimkan masih cukup andal hingga pada baud rate 9600.

Pengujian program komunikasi dilanjutkan dengan pengiriman data - data karakter secara dua arah (*half duplex*), sehingga dapat dilakukan saling transfer data

pada saat yang tidak bersamaan.

Pada pengujian menggunakan media udara yang memanfaatkan fasilitas pemancar radio FM-Stereo, dilakukan pengubahan baud rate mulai dari 50, 75, 110, 134,5 , 150, 300, 600, dan 1200. Dari pengujian ini terlihat sampai baud rate 300, data yang diterima masih cukup andal. Tetapi untuk baud rate di atas 300, terdapat kesalahan pada data yang diterima. Berdasarkan pengujian ini, maka dipilih baud rate sebesar 300 untuk keperluan pengiriman data lewat udara. Adapun format datanya adalah : lebar data 8 bit, tanpa menggunakan parity dan menggunakan 1 stop bit. Urutan (*listing*) program menggunakan bahasa ASSEMBLY selengkapnya terdapat pada lampiran.

4.2 PENGUJIAN MODULATOR DAN DEMODULATOR FSK

Sebelum dilakukan pengujian menggunakan data digital, terlebih dahulu dilakukan pengukuran frekuensi yang mewakili keadaan 'high' dan frekuensi yang mewakili keadaan 'low' dengan menggunakan *frequency counter* dengan merk Hewlett Packard type 5383A. Pengukuran dilakukan pada dua buah MODULATOR FSK, yang selanjutnya disebut dengan alat I dan alat II, hasil pengukuran tersebut terdapat pada tabel 4-1.

TABEL 4-1

HASIL PENGUKURAN FREKUENSI MODULATOR FSK

	ALAT I	ALAT II
Logika '0'	2180 Hz	2195 Hz
Logika '1'	1196 Hz	1192 Hz

Frekuensi yang mewakili logika '0' diperoleh dengan cara mengatur variable resistor (*multi tuned*) R6. Sedangkan frekuensi yang mewakili logika '1' diperoleh dengan cara mengatur variable resistor R7. Dari hasil pengaturan R6 dan R7 yang optimal dari kedua alat tersebut, diperoleh hasil pengukuran seperti terdapat pada tabel 4-1. Dimana menurut perhitungan pada perencanaan, diinginkan frekuensi sebesar 1200 Hz untuk logika '1' dan frekuensi sebesar 2200 Hz untuk logika '0'. Sehingga jika dibandingkan, maka perbedaan antara hasil pengukuran dan hasil perhitungan adalah tidak terlalu jauh.

Selanjutnya dilakukan pengujian demodulator FSK, yaitu : dengan cara menghubungkan modulator dan demodulator FSK dengan memakai sebuah kabel, sementara keduanya dihubungkan pada komputer menggunakan konektor

DB-25 dengan konfigurasi tanpa Handshaking. Kemudian dilanjutkan dengan pengiriman data berkecepatan 50 bps. Pada pengujian ini dilakukan pengaturan bentuk gelombang FSK pada R9, kesimetrian gelombang FSK pada R8, dan amplitudo gelombang FSK pada R1. Sedangkan pada bagian demodulator FSK, pengaturan dilakukan pada Rx yang berfungsi untuk menentukan frekuensi tengah dari FSK. Pengaturan - pengaturan tersebut mengacu pada ke-valid-an dari data yang diterima pada komputer penerima. Kecepatan pengiriman data dinaikkan terus hingga diperoleh kecepatan maksimal sebesar 300 bps, karena dengan kecepatan di atas 300 bps timbul kesalahan pada data yang diterima. Sehingga kecepatan pengiriman dan penerimaan data maksimal dari modulator dan demodulator FSK ini adalah 300 bps.

4.3 PENGUJIAN SISTEM KESELURUHAN

Pada pengujian sistem secara keseluruhan, pin-pin konektor DB-25 dihubungkan dengan konfigurasi untuk hubungan tanpa handshaking. Pin 4 dihubungkan dengan pin 5. Pin 6, pin 8, dan pin 20 saling berhubungan. Pin 2 (TX) dihubungkan ke output modulator FSK, sedangkan pin 3 (RX) dihubungkan ke input demodulator FSK. Sedangkan pin 7 (*Signal Ground*) dihubungkan dengan ground dari peralatan modulator dan demodulator FSK. Konfigurasi ini

adalah identik antara komputer I dan komputer II.

Selanjutnya ferit pada lilitan L4 dari oscillator pemancar FM-Stereo diatur untuk menentukan frekuensi pembawa dari pemancar tersebut. Untuk mengetahui bahwa pemancar tersebut telah bekerja pada band pemancar komersial antara 88 sampai 108 MHz, digunakan radio penerima FM-Stereo.

Langkah berikutnya adalah menghubungkan output dari modulator FSK dengan input audio dari pemancar, dimana hanya dipilih salah satu input, yaitu input *Left* atau input *Right* audio.

Pada radio penerima FM-Stereo dilakukan pengaturan pada bagian oscilltor (*tuning*), sehingga diperoleh frekuensi dari pemancar dengan tepat (*matched*). Hal ini dapat diketahui dari suara mendengung yang ditimbulkan pada speaker. Kemudian pada bagian output untuk earphone dihubungkan dengan input dari demodulator FSK.

Setelah masing-masing unit terhubung satu dengan lainnya seperti telah dijelaskan di atas, sistem tersebut telah siap untuk diuji. Mula-mula unit modulator FSK, demodulator FSK, pemancar dan penerima FM-Stereo diaktifkan terlebih dahulu. Kemudian dilanjutkan dengan komputer I dan komputer II. Setelah semua unit pendukung sistem ini aktif, maka dijalankan

program perangkat lunak pada masing-masing komputer.

Program komunikasi yang dijalankan dapat mengirimkan dan menerima data-data berupa karakter. Mula-mula program tersebut diatur agar mempunyai kecepatan pengiriman data sebesar 5Ø bps. Pada saat data karakter tersebut dikirimkan, dilakukan pengaturan pada unit modulator dan demodulator FSK. Adapun pengaturan unit modulator adalah pada bagian variable resistor (*multi turn*) R1 untuk pengaturan amplitudo, variable resistor R8 untuk pengaturan kesimetrian gelombang, dan variable resistor R9 untuk pengaturan bentuk gelombang FSK. Pengaturan yang dilakukan tersebut mengacu pada ke-valid-an dari data yang diterima pada komputer penerima. Ternyata pada baud rate 5Ø bps tidak timbul adanya kesalahan pada data yang diterima, sehingga baud rate dinaikkan sampai timbul kesalahan pada data yang diterima. Pada akhirnya baud rate maksimal yang dapat dicapai adalah sebesar 3ØØ bps. Dan pada baud rate tersebut hasil pengukuran sinyal output modulator FSK menggunakan voltmeter HELES model SP-2ØD, terdapat pada tabel 4-2.

Selain pengaturan pada unit modulator tersebut, dilakukan pula pengaturan pada unit demodulator dan unit radio penerima. Pengaturan pada unit demodulator dilakukan pada variable resistor Rx, untuk menepatkan

frekuensi tengah dari demodulator tersebut. Sedangkan pada radio penerima adalah dengan mengatur volume suara pada speaker / output earphone, sehingga tidak timbul kesalahan pada data yang diterima pada baud rate maksimal yang dapat dicapai. Hasil pengukuran output tegangan dari speaker / earphone radio yang dipakai untuk baud rate 300 bps, dengan menggunakan oscilloscope merk LEADER type 1041 terdapat pada tabel 4-3.

TABEL 4-2

HASIL PENGUKURAN TEGANGAN SINYAL MODULATOR FSK

	ALAT I	ALAT II
V rms	10 Volt	9,2 Volt
V max	14,14 Volt	13,01 Volt

TABEL 4-3

HASIL PENGUKURAN TEGANGAN OUTPUT PEAK-TO-PEAK DARI SPEAKER RADIO PENERIMA FM-STEREO

	JENSONIC JT-4058S	TENS MW-S6N
Minimal	1,05 Volt	0,26 Volt
Maksimal	6 Volt	5 Volt

BAB V
PENUTUP

5.1 KESIMPULAN

Setelah melalui serangkaian uji coba, maka dapat disimpulkan beberapa hal, yakni :

1. Alat pengirim data digital melalui pemancar radio FM-Stereo ini dapat dioperasikan untuk komputer IBM PC-XT atau IBM PC-AT.
2. Perangkat lunak pendukung komunikasi data dapat menggunakan berbagai bahasa pemrograman, baik bahasa tingkat tinggi maupun bahasa tingkat rendah.
3. Untuk mendapatkan hasil yang optimal, pada program komunikasi data perlu dipilih baud rate dan format data yang tepat antar kedua komputer yang saling berkomunikasi sehingga data-data yang diterima cukup valid.
4. Kualitas data yang diterima ditentukan oleh kualitas dari sistem pemancar dan penerima radio FM-Stereo serta ditentukan pula oleh kondisi cuaca.

5. Alat ini diaplikasikan untuk komunikasi data satu arah (*simplex*) dan dua arah bergantian (*half duplex*).

5.2 SARAN

Berdasarkan pengujian peralatan yang telah dilakukan, maka bukan merupakan hal yang mustahil untuk dilakukan hal-hal berikut sebagai langkah-langkah pengembangan untuk penyempurnaan di masa-masa yang akan datang, diharapkan untuk dilakukan :

1. Pengembangan program perangkat lunak yang dapat mendukung peralatan ini untuk jenis komunikasi dua arah tanpa bergantian (*full duplex*).
2. Pengembangan pengiriman data yang berupa file-file yang dikirimkan antar komputer.
3. Pengembangan perangkat keras agar mempunyai baud rate maksimum yang hampir sama dengan menggunakan kabel, tetapi dengan kesalahan yang relatif kecil.

DAFTAR PUSTAKA

1. Brey, Barry. B., THE INTEL MICROPROCESSORS : 8086/8088, 80186, 80286, 80386, and 80486 Architecture, Programming, and Interfacing, Macmillan Publishing Company, 1991.
2. Coffron, James W., PRACTICAL HARDWARE DETAIL FOR 8080, 8085, Z80, AND 6800 MICROPROSESSOR SYSTEM, New Jersey, Prentice Hall Inc., 1981.
3. Coffron, James W., Z80 APPLICATION, Sybex Inc., 1983
4. Campbell, Joe, THE RS-232 SOLUTION, Second Edition, Campbell Productions, Berkeley, California, 1988.
5. Eggerbrecht, Lewis C., INTERFACING TO IBM PERSONAL COMPUTER, Howard W. Sams & Co. Inc., 1985.
6. Hall, Douglas V., MICROPROCESSORS AND INTERFACING : Programming and Hardware, McGraw-Hill, Inc., 1986.
7. Heijer, P.C. Den, KOMUNIKASI DATA, Elex Media Komputindo, Jakarta, 1991.
8. Held, Gilbert, DATA COMMUNICATION NETWORKING DEVICE, Second Edition, Wiley Corp., 1988.
9. Stallings, William, DATA AND COMPUTERS COMMUNICATION Second Edition, Mac Millian, 1988.

10. Steeman, J.P.M, DATA SHEET BOOK 2, Elex Media Komputindo, Jakarta, 1989.

11. Tomasi, W., FUNDAMENTALS OF ELECTRONIC COMMUNICATIONS SYSTEMS, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

12. Tooley, Michael, DATA COMMUNICATION POCKET BOOK, Heinemann Newnes Ltd., 1989.

13. Uffenbeck, John, THE 8086/8088 FAMILY : DESIGN, PROGRAMMING AND INTERFACING, Prentice Hall Inc., 1987.

15. Wyatt, Allen L., USING ASSEMBLY LANGUAGE, Que Corporation, 1992.

16. _____, MICROSYSTEM COMPONENTS HANDBOOK - MICROPROCESSOR VOLUME 1, Intel Co., 1985.

17. _____, THE IBM PC-XT TECHNICAL REFERENCE

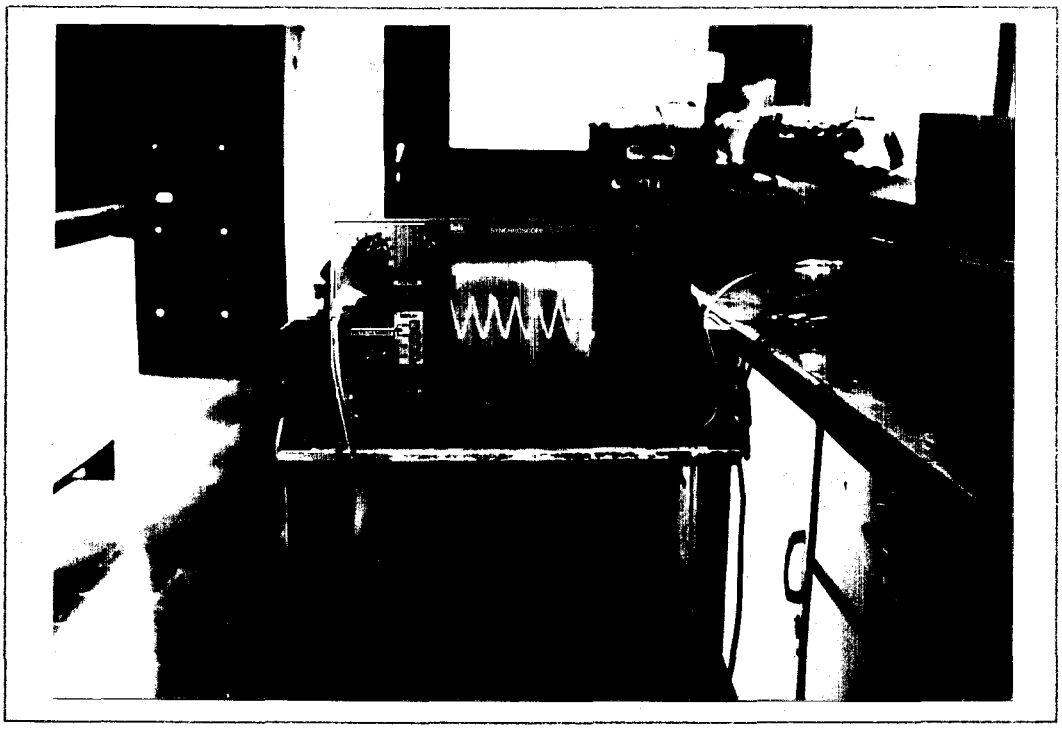


FOTO SINYAL FSK YANG MEWAKILI LOGIKA '1'

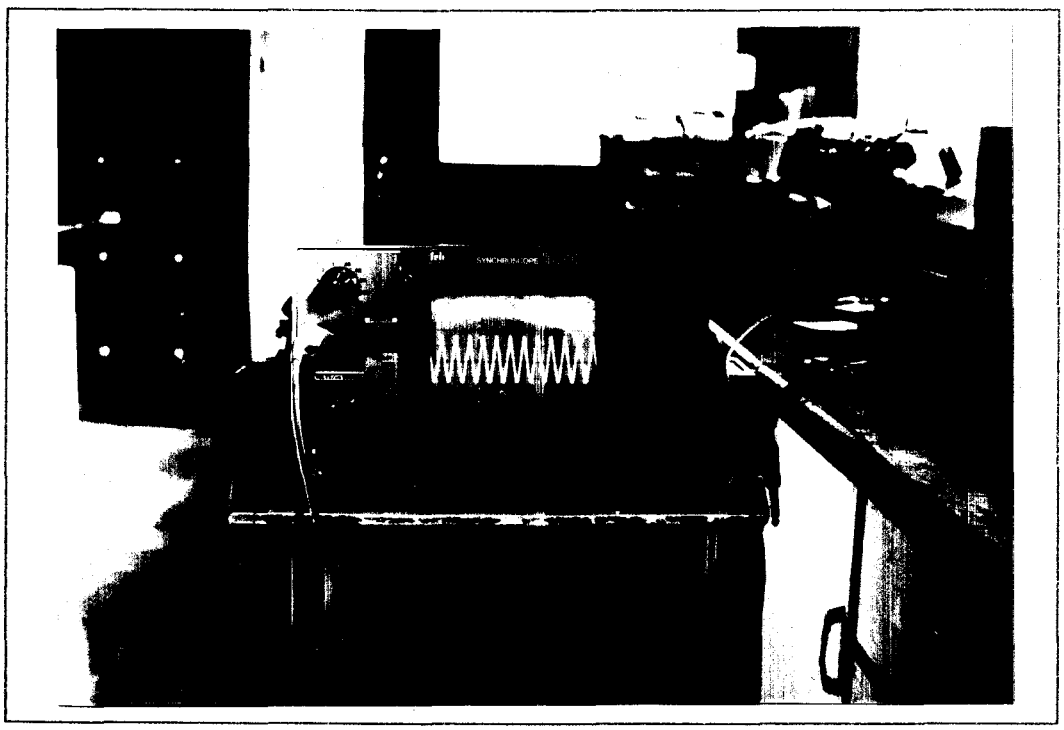


FOTO SINYAL FSK YANG MEWAKILI LOGIKA '0'

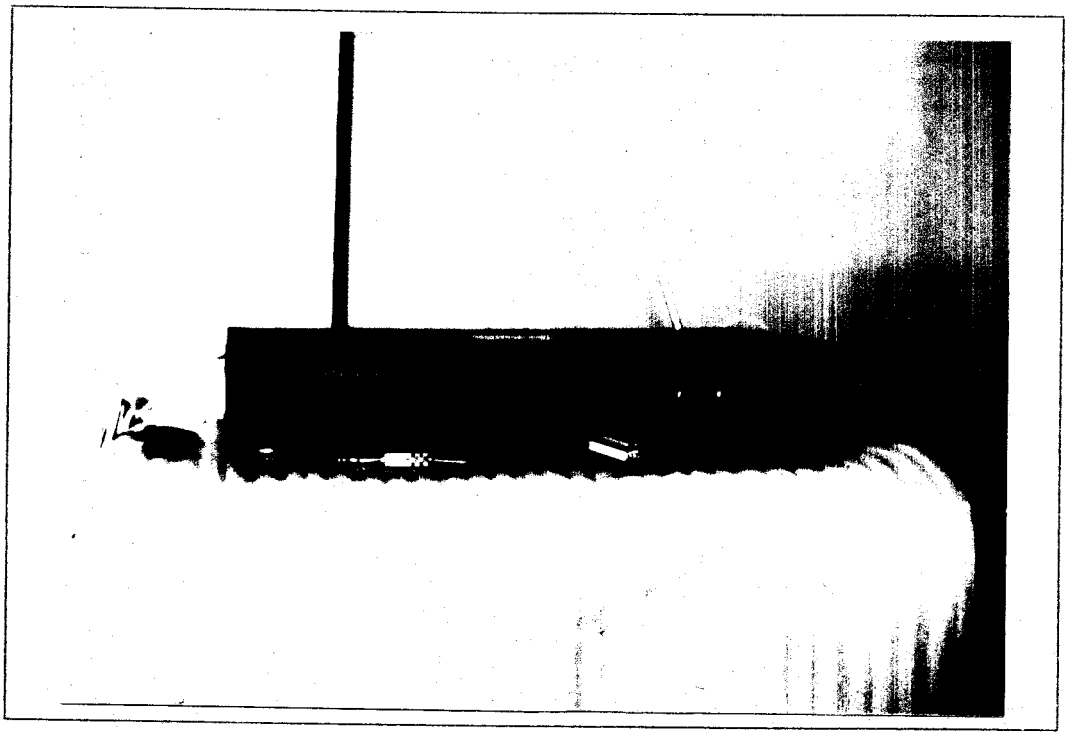
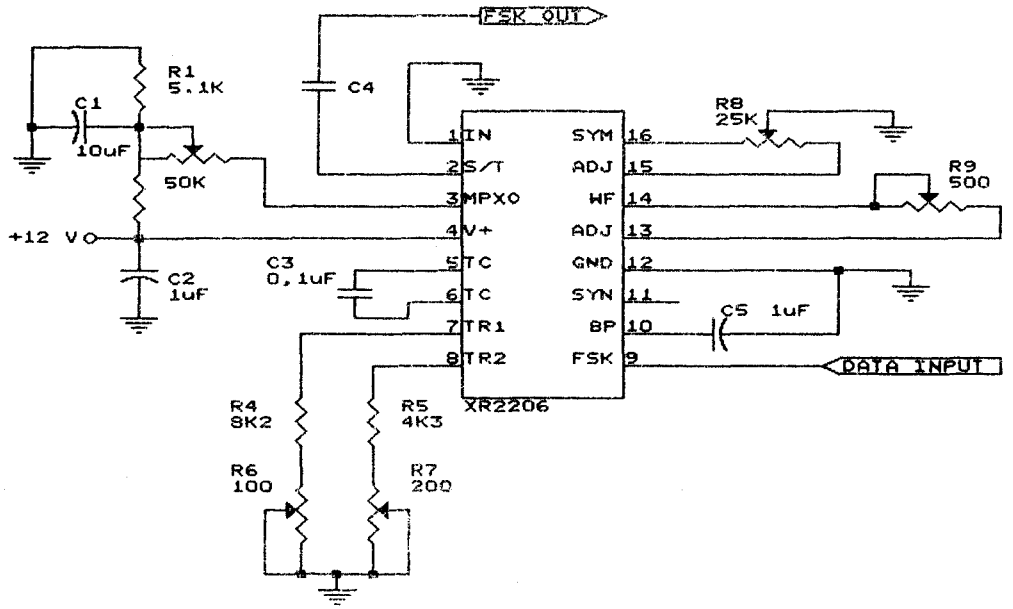
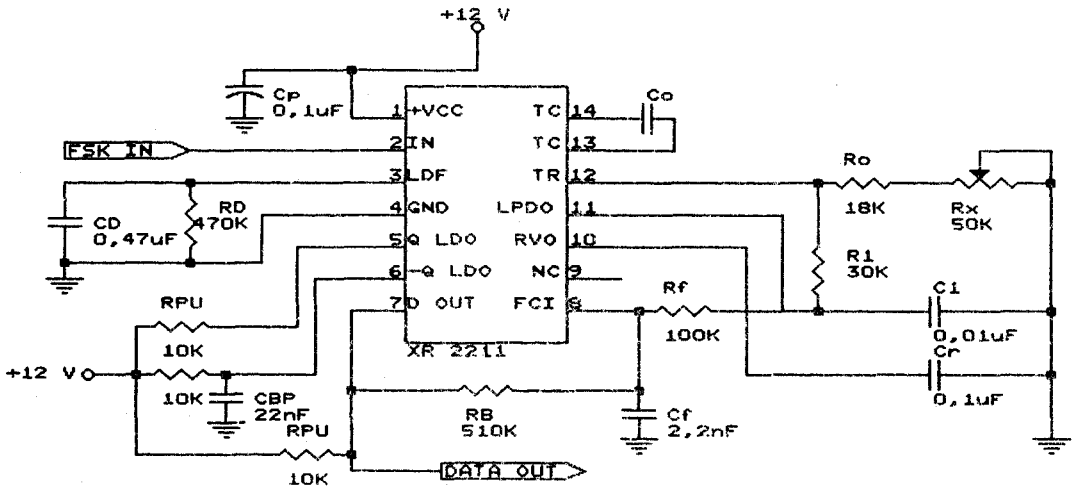


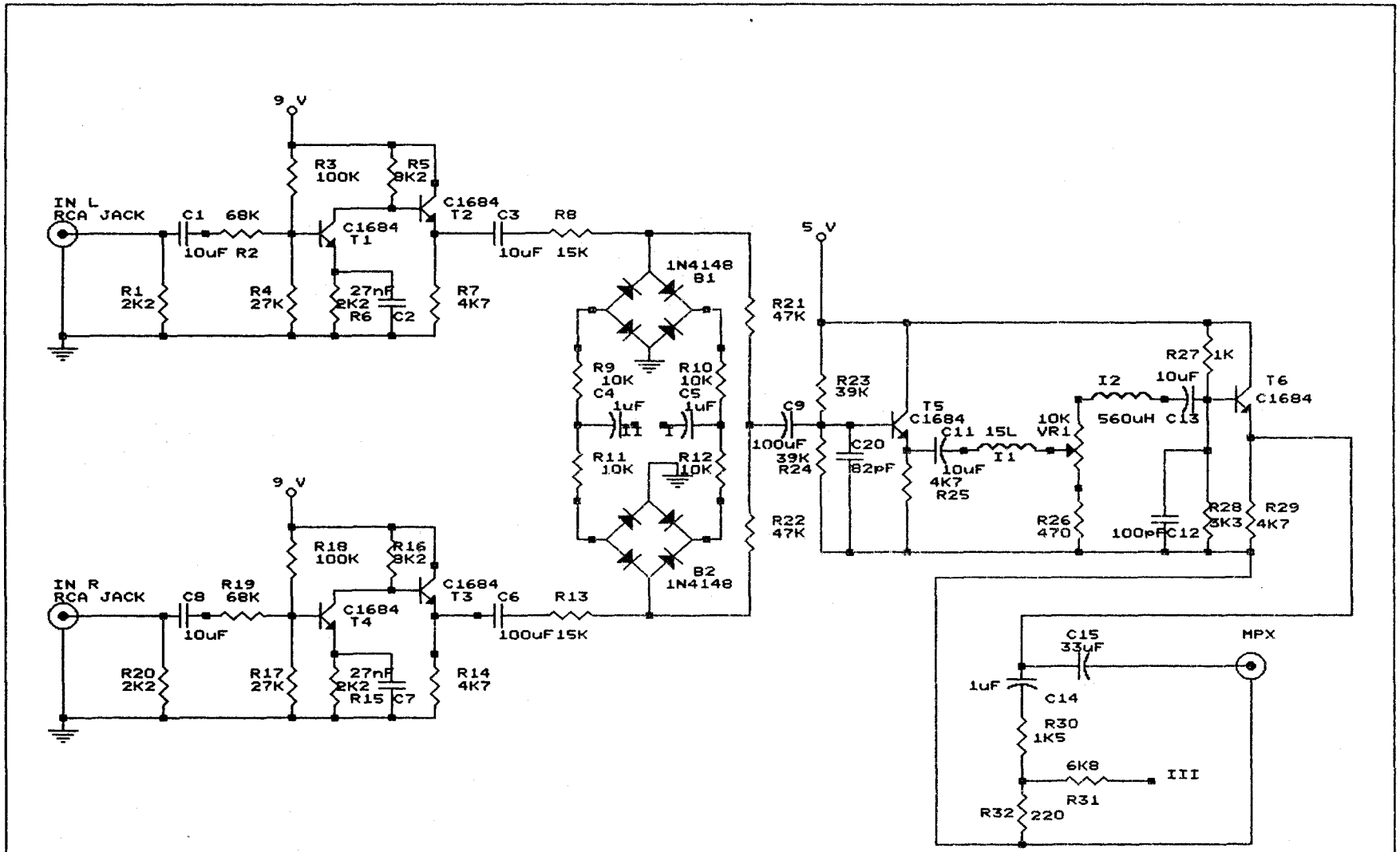
FOTO ALAT



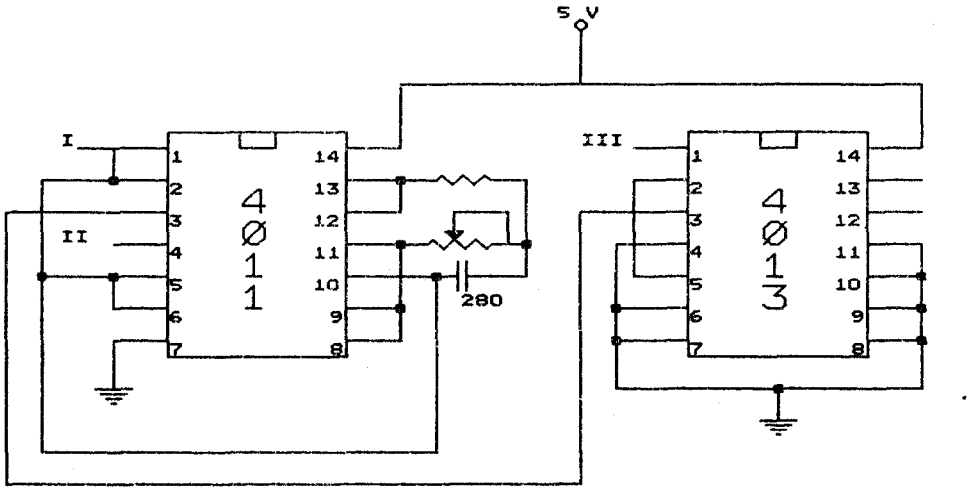
FSK MODULATOR		
Size: A	Document Number: EDWIN ARISTIAWAN (2882200941)	REV
Date: July 12, 1993	Sheet: 5 of 5	



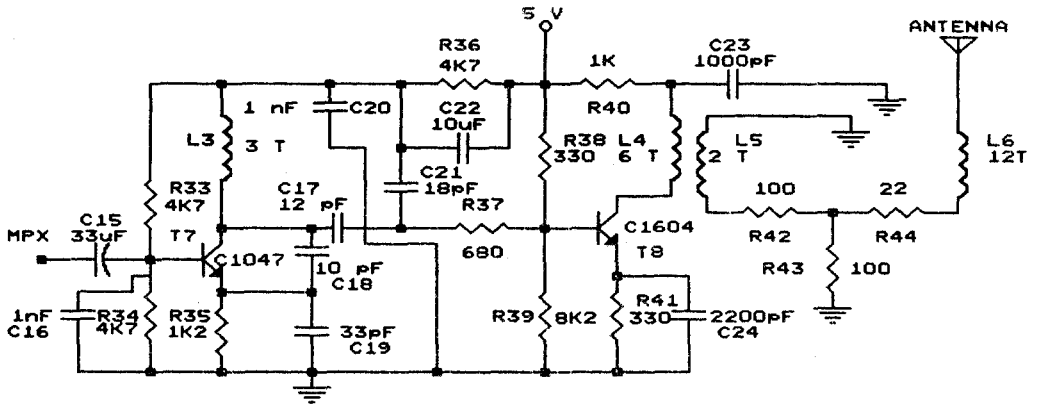
FSK DEMODULATOR		
Size: A	Document Number: EDWIN ARISTIAWAN (2882200941)	REV:
Date: July 12, 1993	Sheet 5 of 5	



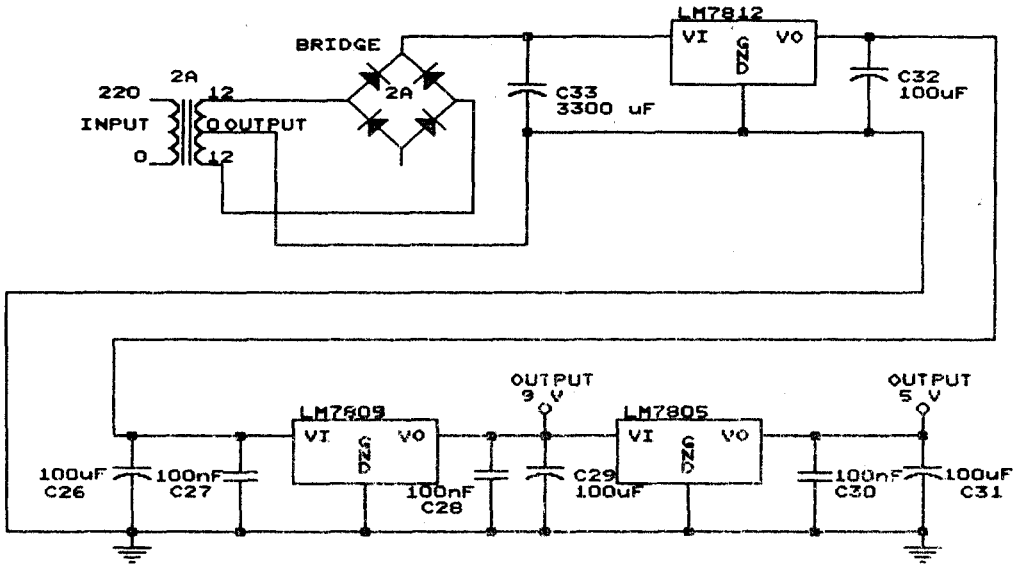
EXCITER FM - STEREO		
Size: A	Document Number: EDWIN ARISTIAWAN (2882200941)	REV:
Date: July 17, 1993	Sheet 1 of 3	



PEMBANGKIT FREKUENSI 19 KHz DAN 38 KHz		
Size: A	Document Number: EDWIN ARISTIAWAN (2882200941)	REV
Date: July 11, 1993	Sheet 3 of 3	



OSCILLATOR FM		
Size: A	Document Number: EDWIN ARISTIAWAN (2882200941)	REV:
Date: July 11, 1993	Sheet 2 of 2	



POWER SUPPLY		
Size: A	Document Number: EDWIN ARISTIAWAN (2882200941)	REV:
Date: July 20, 1993	Sheet 4 of 4	

```
;*****************************************************
;*                  EDWIN  ARISTIAWAN                *
;*                   NRP: 2882200941                 *
;*               TEKNIK ELEKTRO FTI - ITS            *
;*              "  DATA  COMMUNICATION  "            *
;*       PROGRAM SOFTWARE PENDUKUNG TUGAS AKHIR      *
;*****************************************************
;-----------------------------------------------------
; Operasi interupsi waktu menerima data dan melakukan
; total check handshake pada modem controlnya
;-----------------------------------------------------
;Definisi MACRO
;---------------
@dsdata     MACRO
            mov ax,@data
            mov ds,ax
            endm
;
@exe        MACRO
            mov ax,4c00h
            int 21h
            endm
;
@clear      MACRO
            push ax
            mov al,3
            mov ah,0
            int 10h
            pop ax
            endm

@askoo      MACRO
            mov ah,00
            int 16h
            endm

@askoi      MACRO
            mov ah,01h
            int 16h
            endm

@parstr     MACRO bari,kolo,attri,jumst,stri
            mov baris,bari
            mov kolom,kolo
            mov attrib,attri
            mov jumstr,jumst
            mov string,stri
            call Putstr
            endm

@parget     MACRO kolat,kolba,barat,barba,sgmt
            mov kolata,kolat
            mov kolbav,kolba
            mov barata,barat
            mov barbava,barba
            mov offmd,sgmt
```

```
            Call Getimmage
            endm

@parput     MACRO kolat,kolba,barat,barba,atche,sgmt
            mov kolata,kolat
            mov kolbav,kolba
            mov barata,barat
            mov barbava,barba
            mov atchek,atche
            mov offme,sgmt
            Call Putimmage
            endm

@parvin     MACRO kolat,kolba,barat,barba,at,jeni,charfi,attfi
            mov kolata,kolat
            mov kolbav,kolba
            mov barata,barat
            mov barbava,barba
            mov att,at
            mov jenis,jeni
            mov charfil,charfi
            mov attfil,attfi
            call windov
            endm

clrscr      MACRO               ;Clear screen
            mov al,09h          ;menggunakan set video mode 9
            mov ah,00h
            int 10h
            ENDM
;
set_mode    MACRO   mode        ;mode   1   : 40 x 25 color
            mov al,mode         ;        3   : 80 x 25 color
            mov ah,0            ;        4   : 320 x 200 graphic mode
            int 10h
            ENDM
;
gotoyx      MACRO    baris,kolom,page
            mov dh,baris        ;menempatkan kursor ke baris yang
            mov dl,kolom        ;dipilih, ke kolom yang dipilih dan
            mov bh,page         ;ke halaman (page) yang dipilih
            mov ah,2            ;AH <-- function 2 INT 10H
            int 10h
            ENDM
;
readkbd     MACRO               ;Read keyboard  (key in)
            mov ah,0            ;setelah keybord di_penyet
            int 16h             ;karakter ASCII berada di AL
            ENDM                ;scan code berada di AH
;
tul1karv    MACRO
            mov cx,1            ;harus diavali MOV AL,karakter
            mov bh,0            ;              MOV BL,attribute
            mov ah,9
            int 10h
            ENDM
```

```
;
tulikarTTY MACRO
             mov bh,0
             mov ah,0Eh
             int 10h
             ENDM
;
tulis        MACRO tulisan
             mov dx,offset tulisan
             mov ah,09h
             int 21h
             ENDM
;
LF_CR        MACRO
             mov al,0dh
             mov ah,0eh
             int 10h
             mov al,0ah
             mov ah,0eh
             int 10h
             ENDM
;
report       MACRO ro,col,varna,baris,kolom1,kolom2,hlmn
             mov al,col
             mov column,al
             mov al,ro
             mov row,al
             mov al,varna
             mov color,al
             mov al,baris
             mov bar,al
             mov al,kolom1
             mov kol1,al
             mov al,kolom2
             mov kol2,al
             mov al,hlmn
             mov hal,al
             Call Line_col_counter
             ENDM
;
;
; ADRES REGISTER PADA UART INS_8250
;-----------------------------------
TX_BUFFER            EQU 03F8H
RX_BUFFER            EQU 03F8H
BAUD_DIV_LSB         EQU 03F8H
BAUD_DIV_MSB         EQU 03F9H
INT_ENBL_REG         EQU 03F9H
INT_ID_REG           EQU 03FAH
LINE_CONT_REG        EQU 03FBH
MODEM_CONT_REG       EQU 03FCH
LINE_STAT_REG        EQU 03FDH
MODEM_STAT_REG       EQU 03FEH
;
INTA00               EQU 20H
INTA01               EQU 21H
```

```
;
                    DOSSEG
                   .MODEL LARGE
                   .STACK 400H
;
;Tempat DATA
;------------
                   .DATA
judul_1             db 'Tes INS 8250 pada operasi normal,'
                       'dibuat loopback $'
judul_2             db 'dengan menyambung TxD dengan RxD, '
                        RTS dengan CTS $'
judul_3             db 'Jika menerima 1 karakter, langsung '
                       'memberi interupsi$'
percobaan           db 'percobaan --->  $'
JUDUL_4             db '                Ketik_kan karakter apa'
                       'saja $'
JUDUL_5             db '   Ketik tombol " Esc " untuk exit '
                       'dari program ini $'
TUL_TX_NO_EMPTY     db 'TX HOLDING REGISTER tidak pernah '
                       'empty $'
tul_data_NOT_READY  db 'Data tidak pernah diterima $'
DSR_high            db '      pin DSR selalu high ---> '
                       'periksa DTR lavan    $'
CTS_high            db '      pin CTS selalu high ---> '
                       'periksa RTS lavan    $'
press_any_key       db 'Press any key to continue $'
data_kirim          db 'KARAKTER YANG DIKIRIMKAN'
data_terima         db 'KARAKTER YANG DITERIMA  '
data_line_column    db ' Line:    Column:     '
sela                db '                                  $'
selo                db '                $'
L_corner            db 218,213,201,214
R_corner            db 191,184,187,183
BR_corner           db 217,190,188,189
BL_corner           db 192,212,200,211
Horiz               db 196,205,205,196
Verti               db 179,179,186,186
data1               db ' F1=Clear Screen TX   F2=Print '
                       'Screen   F3=BAUD  '
                    db ' F4=Help   Esc=Back to DOS    '
data2               db ' EDWIN ARISTIAWAN Nrp.2882200941,
                       'Bidang Studi
Elektronika, Teknik Elektro - ITS '
data3               db 185,' ALAT PENGIRIM DATA DIGITAL '
                       'DENGAN METODE FSK ',204
data4               db ' PETUNJUK PEMAKAIAN SOFTWARE '
                       'KOMUNIKASI DATA :    '
data5               db ' 1. Sebelum mengirim data harap '
                       'menekan tombol F1.'
data6               db ' 2. F2 digunakan untuk PrintScreen.'
data7               db ' 3. F3 digunakan untuk mengatur '
                       'kecepatan data.   '
data8               db ' 4. Kecepatan TX harus sama dengan '
                       'kecepatan RX.   '
```

```
data9           db' 5. Kalau mengalami kesulitan harap'
                  ' menghubungi : '
data10          db'"LAB B-205 TEKNIK ELEKTRO FTI-ITS"'
data11          db'                                  '
data12          db'                          Edwin A.R. '
Pre4            db'            TUGAS AKHIR           '
Pre5            db'  Copyright (c) 1993 by '
                  'EDWIN ARISTIAWAN  '
batas           db198,78 dup (205),181
choice1         db'F1=Clear Screen TX'
choice2         db'F2=Print Screen'
choice3         db'F3=BAUD'
choice4         db'F4=Help'
writefl         db' Pilih Tingkat Bps :  '
newfill         db' [ 1, 2, 3, 4, 5, 6 ] '
PROMPT          db'B> $'
row_k           db1
row_t           db13
column_k        db0
column_t        db0
page_k          db0
page_t          db0
row             db?
column          db?
color           db?
bar             db?
kol1            db?
kol2            db?
hal             db?
BACKSPACE       db?
BLANK           db?
CR              db?
ARROW           db?
ARROW_RX        db?
RGT_ARROW       db?
RGT_ARROW_RX    db?
UP_ARROW        db?
UP_ARROW_RX     db?
DOWN_ARROW      db?
DOWN_ARROW_RX   db?
HOME            db?
HOME_RX         db?
END_TX          db?
END_RX          db?
PG_UP           db?
PG_UP_RX        db?
PG_DOWN         db?
PG_DOWN_RX      db?
CLS             db?
CLS_R           db?
KOORDY_BAR1     db?
KOORDY_BAR2     db?
KOORDY_BAR3     db?
KOORDY_BAR4     db?
KOORDY_BAR5     db?
KOORDY_BAR6     db?
```

```
KOORDY_BAR7             db ?
KOORDY_BAR8             db ?
KOORDY_BAR9             db ?
KOORDY_BAR10            db ?
KOORDY_BAR11            db ?
KOORDY_BAR13            db ?
KOORDY_BAR14            db ?
KOORDY_BAR15            db ?
KOORDY_BAR16            db ?
KOORDY_BAR17            db ?
KOORDY_BAR18            db ?
KOORDY_BAR19            db ?
KOORDY_BAR20            db ?
KOORDY_BAR21            db ?
KOORDY_BAR22            db ?
KOORDY_BAR23            db ?
BARATAS                 db ?
BARBAW                  db ?
WARNA                   db ?
CLS_RX                  db ?
PRT                     db ?
BPS_MS                  db ?
BPS_LS                  db ?
offme                   dw ?
offmd                   dw ?
cacah_huruf             dw 0
address_offset          dw 0
atchek                  dw ?
ceker                   dw ?
baris                   dw ?
kolom                   dw ?
attrib                  dw ?
jumstr                  dw ?
string                  dw ?
many                    dw ?
offmen1                 db 2*1024 dup (?)
offmen2                 db 2*1024 dup (?)
offmen5                 db 2*1024 dup (?)
offmen6                 db 2*1024 dup (?)
kolata                  db ?
kolbaw                  db ?
barata                  db ?
barbawa                 db ?
att                     db ?
jenis                   db ?
charfil                 db ?
attfil                  db ?
clear                   db ?
tas                     db ?
simpanan_isi_BL         db ?
attribute               db ?
simpanan_huruf          db 40H
adres_INT_OCH           dd 0
;
                        .CODE
PROGRAM                 PROC
```

```
                    mov BPS_MS,01h
                    mov BPS_LS,080h
INIS:
                    Call INITIATE
                    clrscr
                    gotoyx 0,0,0
                    tulis judul_1
                    LF_CR
;
                    tulis judul_2
                    LF_CR
;
                    tulis judul_3
                    LF_CR
                    LF_CR
                    tulis percobaan
;
                    Call Data_out_1
;
                    LF_CR
                    LF_CR
                    tulis judul_4
                    LF_CR
                    tulis judul_5
                    LF_CR
                    tulis press_any_key

WAITING:            mov ah,01h
                    int 16h
                    jz WAITING
WALL:
                    Call Disply
                    Call Setting_kur
;
TES_KIRIM:
                    Call Handshake
;
DATA_KEYBOARD:
                    report row_k,column_k,060h,2.52,63,page_k
                    gotoyx row_k,column_k,page_k
                    READKBD         ;Tes kirim huruf menggunakan
                                    ;keyboard
KYBD:               cmp al,01bh     ;01BH : ASCII code tombol Esc
                    je EXIT_1A
                    cmp ah,60
                    jnz CHECK_F3
                    @parstr 0,0,56,80,<offset data1>
                    @parstr 0,22,15,15,<offset choice2>
                    @askoo
                    cmp al,13
                    jne CHECK_F3
                    Call PrtScr
                    jmp TES_KIRIM
EXIT_1A:
                    jmp EXIT_1B
CHECK_F3:
```

```
                    cmp ah,61
                    jnz CHECK_F4
                    @parstr 0,0,56,80,<offset data1>
                    @parstr 0,40,15,7,<offset choice3>
                    @askoo
                    cmp al,13
                    je HIA
                    jmp KYBD
EXIT_1B:            jmp EXIT_1C
HIA:
                    Call Print
                    jmp KYBD
CHECK_F4:           cmp ah,62
                    jnz CHECK_F1A
                    @parstr 0,0,56,80,<offset data1>
                    @parstr 0,50,15,7,<offset choice4>
                    @askoo
                    cmp al,13
                    je HIO
                    jmp KYBD
CHECK_F1A:          jmp CHECK_F1B
HIO:
                    @parget 11,65,1,13,<offset offmen1>
                    @parget 13,65,2,13,<offset offmen2>
                    @parput 13,65,2,13,7,<offset offmen2>
                    @parwin 11,62,1,11,64,1,0,64
                    @parstr 2,12,78,50,<offset data4>
                    jmp SINU
CHECK_F1B:          jmp CHECK_F1
EXIT_1C:            jmp EXIT_1D
SINU:               @parstr 3,12,78,50,<offset data5>
                    @parstr 4,12,78,50,<offset data6>
                    @parstr 5,12,78,50,<offset data7>
                    @parstr 6,12,78,50,<offset data8>
                    jmp SUNI
EXIT_1D:            jmp EXIT_1E
SUNI:               @parstr 7,12,78,50,<offset data9>
                    @parstr 8,12,78,50,<offset data10>
                    @parstr 9,12,78,50,<offset data11>
                    @parstr 10,12,78,5C,<offset data12>
                    @askoo
                    @parput 11,65,1,13,0,<offset offmen1>
                    jmp KYBD
EXIT_1E:            jmp EXIT_1
CHECK_F1:
                    cmp ah,59
                    jnz CHECK_END_TX_1
                    mov CLS,1
                    mov PG_UP,0
                    mov PG_DOWN,0
                    mov HOME,0
                    mov END_TX,0
                    mov DOWN_ARROW,0
                    mov UP_ARROW,0
                    mov RGT_ARROW,0
                    mov ARROW,0
```

```
                        @parstr 0,0,56,80,<offset data1>
                        @parstr 0,1,15,18,<offset choice1>
                        jmp BY
CHECK_END_TX_1:     jmp CHECK_END_TX
BY:
                        @askoo
                        cmp al,13
                        je WERR
                        jmp KYBD
WERR:
                        Call Block
                        jmp NOT_BACK
EXIT_1:               JMP EXIT_2
CHECK_END_TX:
                        @parstr 0,0,56,80,<offset data1>
                        mov CLS,0
                        cmp ah,79
                        jnz CHECK_PG_UP
                        mov al,16
                        mov PG_UP,0
                        mov PG_DOWN,0
                        mov HOME,0
                        mov END_TX,1
                        mov DOWN_ARROW,0
                        mov UP_ARROW,0
                        mov RGT_ARROW,0
                        mov ARROW,0
                        jmp NOT_BACK
CHECK_PG_UP:
                        mov END_TX,0
                        cmp ah,73
                        jnz CHECK_PG_DOWN
                        mov al,30
                        mov PG_UP,1
                        mov PG_DOWN,0
                        mov HOME,0
                        mov DOWN_ARROW,0
                        mov UP_ARROW,0
                        mov RGT_ARROW,0
                        mov ARROW,0
                        jmp NOT_BACK
CHECK_PG_DOWN:
                        mov PG_UP,0
                        cmp ah,81
                        jnz CHECK_HOME
                        mov al,31
                        mov PG_DOWN,1
                        mov HOME,0
                        mov DOWN_ARROW,0
                        mov UP_ARROW,0
                        mov RGT_ARROW,0
                        mov ARROW,0
                        jmp NOT_BACK
EXIT_2:               JMP EXIT_3
CHECK_HOME:
                        mov PG_DOWN,0
```

```
                    cmp ah,71
                    jnz CHECK_D_ARROW
                    mov al,17
                    mov HOME,1
                    mov DOWN_ARROW,0
                    mov UP_ARROW,0
                    mov RGT_ARROW,0
                    mov ARROW,0
                    jmp NOT_BACK
CHECK_D_ARROW:
                    mov HOME,0
                    cmp ah,80
                    jnz CHECK_U_ARROW
                    mov al,25
                    mov DOWN_ARROW,1
                    mov UP_ARROW,0
                    mov RGT_ARROW,0
                    mov ARROW,0
                    jmp NOT_BACK
CHECK_U_ARROW:
                    mov DOWN_ARROW,0
                    cmp ah,72
                    jnz CHECK_R_ARROW
                    mov al,24
                    mov UP_ARROW,1
                    mov RGT_ARROW,0
                    mov ARROW,0
                    jmp NOT_BACK
EXIT_3:             JMP EXIT_4
CHECK_R_ARROW:
                    mov UP_ARROW,0
                    cmp ah,77
                    jnz CHECK_L_ARROW
                    mov al,26
                    mov RGT_ARROW,1
                    mov ARROW,0
                    jmp NOT_BACK
CHECK_L_ARROW:
                    mov RGT_ARROW,0
                    cmp ah,75
                    jnz CHECK_BSPCE
                    mov al,29
                    mov ARROW,1
                    jmp NOT_BACK
CHECK_BSPCE:
                    mov ARROW,0
                    cmp al,08h
                    jnz NOT_BACK
                    mov BACKSPACE,1
                    jmp APPEAR
NOT_BACK:
                    mov BACKSPACE,0
                    cmp al,13
                    je MASUK
                    mov CR,0
                    jmp APPEAR
```

```
MASUK:
                    mov CR,1
                    Call Enter_TX
APPEAR:
                    mov dx,TX_BUFFER
                    out dx,al
                    cmp ARROW,1
                    jne ENTHER
                    jmp KURSOR_KIRI
ENTHER:
                    cmp CR,1
                    jne U_ARROW
                    jmp KURSOR_KIRI
U_ARROW:            cmp UP_ARROW,1
                    jne D_ARROW
                    dec row_k
                    cmp row_k,2
                    jnz PASS
                    mov row_k,3
PASS:
                    jmp ATUR_POS_KUR
EXIT_4:             JMP EXIT_5
;
D_ARROW:            cmp DOWN_ARROW,1
                    jne PG_DOWN_TX
                    inc row_k
                    cmp row_k,11
                    jle PASS2
                    mov row_k,11
PASS2:
                    jmp ATUR_POS_KUR
;
PG_DOWN_TX:
                    cmp PG_DOWN,1
                    jne PG_UP_TX
                    mov row_k,11
                    jmp ATUR_POS_KUR
;
PG_UP_TX:
                    cmp PG_UP,1
                    jne HOME_TX
                    mov row_k,3
                    jmp ATUR_POS_KUR
;
HOME_TX:            cmp HOME,1
                    jne AKHIR_TX
                    mov column_k,1
                    jmp ATUR_POS_KUR
;
AKHIR_TX:           cmp END_TX,1
                    jne BSPCE
                    mov dh,row_k
                    Call Compare_pos_bar
                    mov column_k,dl
                    jmp ATUR_POS_KUR
;
```

```
EXIT_5:             JMP EXIT_6
BSPCE:
                    cmp BACKSPACE,1
                    jnz STRIGHT
                    mov al,' '
                    jmp KURSOR_KIRI
KURSOR_KIRI:
                    Call Kursor_mundur_TX
ATUR_POS_KUR:
                    GOTOYX row_k,column_k,page_k
STRIGHT:
                    cmp CLS,1
                    je SAVE_POS
                    cmp END_TX,1
                    je SAVE_POS
                    cmp PG_UP,1
                    je SAVE_POS
                    cmp PG_DOWN,1
                    je SAVE_POS
                    cmp HOME,1
                    je SAVE_POS
                    cmp DOWN_ARROW,1
                    je SAVE_POS
                    cmp UP_ARROW,1
                    je SAVE_POS
                    cmp RGT_ARROW,1
                    je SAVE_POS
                    cmp ARROW,1
                    je SAVE_POS
                    cmp CR,1
                    jne HURUF
                    mov al,' '
HURUF:
                    mov dl,column_k
                    mov dh,row_k
                    Call Simpan_koor_kur
                    mov bl,01eh
                    TUL1KARW
                    jmp SAVE_POS
EXIT_6:             jmp EXIT
SAVE_POS:
                    cmp BACKSPACE,1
                    jz LOOK
                    cmp ARROW,1
                    jz LOOK
                    cmp UP_ARROW,1
                    jz BACA_POS_KUR
                    cmp DOWN_ARROW,1
                    jz BACA_POS_KUR
                    cmp HOME,1
                    jz BACA_POS_KUR
                    cmp PG_DOWN,1
                    jz BACA_POS_KUR
                    cmp PG_UP,1
                    jz BACA_POS_KUR
                    cmp END_TX,1
```

```
                        jz BACA_POS_KUR
                        cmp column_k,77
                        jnz MAJUU
                        Call Check_space_TX
MAJUU:
                        Call Kursor_maju_TX
LOOK:
                        cmp CLEAR,1
                        jne BACA_POS_KUR
                        GOTOYX 3,1,0
BACA_POS_KUR:
                        mov ah,03h
                        int 10h
                        mov row_k,dh
                        mov column_k,dl
                        mov page_k,bh

                        jmp TES_KIRIM
EXIT:
                        clrscr
;
                        mov ax,@DATA
                        mov ds,ax
;
                        mov si,offset adres_INT_0CH
                        mov bx, word ptr ds:[si]
                        mov es, word ptr ds:[si+2]
;
                        mov ax,0000H
                        mov ds,ax
;
                        mov si,4*0ch
                        mov word ptr ds:[si],bx
                        mov word ptr ds:[si+2],es
;
                        mov ax,4c00h
                        int 21h
;-------------------------------------------------------------
:Interrupt Service Routine
 INTR_SERVIS:       STI
                        PUSH DS
                        PUSH ES
                        PUSH DI
                        PUSH SI
                        PUSH BP
                        PUSH AX
                        PUSH BX
                        PUSH CX
                        PUSH DX
;
                        MOV DX,INT_ID_REG
                        IN AL,DX
                        TEST AL,04H
                        JNZ IN_DATA_INT_0C
                        JMP EXIT_INT_0C
;
```

```
IN_DATA_INT_OC:
IN_DATA:
                        mov dx,RX_BUFFER
                        in al,DX
                        cmp al,08H
                        jnz CHAR
                        mov ARROW_RX,0
                        mov RGT_ARROW_RX,0
                        mov UP_ARROW_RX,0
                        mov DOWN_ARROW_RX,0
                        mov PG_DOWN_RX,0
                        mov PG_UP_RX,0
                        mov HOME_RX,0
                        mov END_RX,0
                        mov BLANK,1
                        mov al,' '
                        jmp MINUS_BAR
CHAR:
                        mov BLANK,0
                        cmp al,29
                        jnz CHECK_R_ARROW2
                        mov ARROW_RX,1
                        mov RGT_ARROW_RX,0
                        mov UP_ARROW_RX,0
                        mov DOWN_ARROW_RX,0
                        mov PG_DOWN_RX,0
                        mov PG_UP_RX,0
                        mov HOME_RX,0
                        mov END_RX,0
                        jmp MINUS_BAR_2
CHECK_R_ARROW2:
                        mov ARROW_RX,0
                        cmp al,26
                        jnz CHECK_UP_ARROW2
                        mov RGT_ARROW_RX,1
                        mov UP_ARROW_RX,0
                        mov DOWN_ARROW_RX,0
                        mov PG_DOWN_RX,0
                        mov PG_UP_RX,0
                        mov HOME_RX,0
                        mov END_RX,0
                        jmp langsung
CHECK_UP_ARROW2:
                        mov RGT_ARROW_RX,0
                        cmp al,24
                        jnz CHECK_DOWN_ARROW2
                        mov UP_ARROW_RX,1
                        mov DOWN_ARROW_RX,0
                        mov PG_DOWN_RX,0
                        mov PG_UP_RX,0
                        mov HOME_RX,0
                        mov END_RX,0
                        dec row_t
                        cmp row_t,13
                        jnz GO_AHEAD
                        mov row_t,14
```

```
GO_AHEAD:
                        jmp langsung
CHECK_DOWN_ARROW2:
                        mov UP_ARROW_RX,0
                        cmp al,25
                        jnz CHECK_HOME_RX
                        mov DOWN_ARROW_RX,1
                        mov PG_DOWN_RX,0
                        mov PG_UP_RX,0
                        mov HOME_RX,0
                        mov END_RX,0
                        inc row_t
                        cmp row_t,22
                        jle GO_AHEAD2
                        mov row_t,22
GO_AHEAD2:
                        jmp langsung
CHECK_HOME_RX:
                        mov DOWN_ARROW_RX,0
                        cmp al,17
                        jnz CHECK_END_RX
                        mov HOME_RX,1
                        mov END_RX,0
                        mov PG_DOWN_RX,0
                        mov PG_UP_RX,0
                        mov column_t,1
                        jmp langsung
CHECK_END_RX:
                        mov HOME_RX,0
                        cmp al,16
                        jnz CHECK_PG_DN_RX
                        mov END_RX,1
                        mov PG_DOWN_RX,0
                        mov PG_UP_RX,0
                        mov dh,row_t
                        Call Compare_pos_barRX
                        mov column_t,dl
                        jmp langsung
CHECK_PG_DN_RX:
                        mov END_RX,0
                        cmp al,31
                        jnz CHECK_PG_UP_RX
                        mov PG_DOWN_RX,1
                        mov PG_UP_RX,0
                        mov row_t,22
                        jmp langsung
CHECK_PG_UP_RX:
                        mov PG_DOWN_RX,0
                        cmp al,30
                        jnz CHECK_RESIK
                        mov PG_UP_RX,1
                        mov row_t,14
                        jmp langsung
CHECK_RESIK:
                        mov PG_UP_RX,0
                        cmp al,01h
```

```
                    jnz CHECK_CR_2
                    mov CLS_RX,1
                    mov BARATAS,14
                    mov BARBAW,22
                    mov WARNA,01fh
                    Call Bersih
                    mov row_t,14
                    mov column_t,1
                    mov page_t,0
                    jmp langsung
CHECK_CR_2:
                    mov CLS_RX,0
                    push ax
                    cmp al,13
                    jne LANGSUNG
                    pop ax
                    Call Enter_RX
                    push ax
                    jmp MINUS_BAR_2
LANGSUNG:
                    jmp STAY
MINUS_BAR:
                    push ax
                    cmp BLANK,1
                    jnz STAY
MINUS_BAR_2:
                    Call Kursor_mundur_RX
STAY:
                    GOTOYX ROW_T,COLUMN_T,PAGE_T
                    cmp ARROW_RX,1
                    jz LOOK2
                    cmp RGT_ARROW_RX,1
                    jz SAVE_POS1
                    cmp UP_ARROW_RX,1
                    jz READ_POS_KUR
                    cmp DOWN_ARROW_RX,1
                    jz READ_POS_KUR
                    cmp HOME_RX,1
                    jz READ_POS_KUR
                    cmp PG_DOWN_RX,1
                    jz READ_POS_KUR
                    cmp PG_UP_RX,1
                    jz READ_POS_KUR
                    cmp END_RX,1
                    jz READ_POS_KUR
                    cmp CLS_RX,1
                    jz READ_POS_KUR
                    pop ax
                    mov dl,column_t
                    mov dh,row_t
                    Call Simpan_koor_kurRX
                    mov bl,01fh
                    TUL1KARW
SAVE_POS1:
                    cmp BLANK,1
                    jz LOOK2
```

```
                        cmp column_t,77
                        jnz ONWARD
                        Call Check_space_RX
ONWARD:
                        Call Kursor_maju_RX
LOOK2:
                        cmp CLEAR,1
                        jne READ_POS_KUR
                        GOTOYX 14,1,0
READ_POS_KUR:
                        mov ah,03h
                        int 10h
                        mov row_t,dh
                        mov column_t,dl
                        mov page_t,bh
                        report row_t,column_t,060h,13,52,63,page_t
;
EXIT_INT_OC:        mov dx,INT_ENBL_REG
                        mov al,01h
                        out dx,al
;
                        mov al,20h
                        out INTA00,al
;
                        POP DX
                        POP CX
                        POP BX
                        POP AX
                        POP BP
                        POP SI
                        POP DI
                        POP ES
                        POP DS
                        IRET
;
PROGRAM             ENDP
;-------------------------------------------------------------

;------Procedure Set Vektor Addres & Inisialisasi 8250-----
;Mengutip vektor adres INT OCH, asli
;menggunakan INT 21H function 35H pada DOS function call
;Hasil kutipan berada di register ES dan BX
INITIATE            Proc Near
                        CLI
                        MOV AL,0CH
                        MOV AH,35H
                        INT 21H
;
;Vektor adres tersebut dikutipkan disimpan di adres_INT_OCH
                        MOV AX,@DATA
                        MOV DS,AX

                        MOV SI,offset adres_INT_OCH
                        MOV WORD PTR DS:[SI],BX
                        MOV WORD PTR DS:[SI+2],ES
;
```

```
                        MOV DX,CS       ; menaruh adres int_servis
                                        ; routine
                        MOV DS,DX       ; 2000:0000 di lokasi
                                        ; 0000:0030
                        LEA DX,INTR_SERVIS; untuk melayani
                                        ; INT 0CH.
                        MOV AL,0CH      ; Cara menaruh menggunakan
                        MOV AH,25H      ; SOFTWARE INT 21H AH=25H.
                        INT 21H
;
                        MOV AX,@DATA
                        MOV DS,AX
                        MOV ES,AX
;
;Inisialisasi    INS_8250:    7_bit    data,    parity    disable,    300
;baud, 1_stop bit.
                        MOV DX,LINE_CONT_REG
                        MOV AL,10000000B   ; bit_7 pada
                                           ;INT_CONT_REG diberi 1_
                        OUT DX,AL       ; untuk menggarap BAUD_DIV_REG
;
                        MOV DX,BAUD_DIV_MSB   ;Clock pada UART_8250
                                              ;yang ada
                        MOV AL,BPS_MS         ;didalam = 1.843 MHz.
                        OUT DX,AL
;
                        MOV DX,BAUD_DIV_LSB   ; diperlukan divisor
                                              ; 0180H untuk
                        MOV AL,BPS_LS         ; mendapatkan baud
                                              ; rate 300.
                        OUT DX,AL             ; angka LSB = 80H,
                                              ;MSB = 01H.
                        MOV DX,LINE_CONT_REG
                        MOV AL,00000011B      ;8 bit, parity disa-
                        OUT DX,AL             ;ble, 1 stop bit
;
                        MOV DX,INT_ENBL_REG
                        MOV AL,01h
                        OUT DX,AL
;
                        MOV AL,10100100B      ;IRQ4 = bit 4 =
                                              ;diberi 0 = untuk
                        OUT INTA01,AL         ;membuka jalur
                                              ;interupsi IRQ4 8259
                        MOV CX,64
baca_RX_BUFFER:     MOV DX,RX_BUFFER
                        IN AL,DX
                        LOOP baca_RX_BUFFER
;
                        STI
                        ret
INITIATE                endp

;----------Procedure Handshake-------------------
Handshake               Proc Near
                        mov dx,MODEM_CONT_REG
```

```
                    mov al,00001111b        ; DTR <= 1
                    out dx,al
;
                    mov cx,10000
TES_DSR:            mov dx,MODEM_STAT_REG
                    in al,dx
                    and al,00100000B        ; tes DSR
                    jnz TES_CTS
                    loop TES_DSR
;
                    LF_CR
                    tulis DSR_high          ; DSR = 0 berarti
                                            ;pin DSR high
                    LF_CR
;
                    mov dx,MODEM_CONT_REG
                    mov al,00001111B        ; RTS <= 1
                    out dx,al
;
                    mov cx,10000
TES_CTS:            mov dx,MODEM_STAT_REG
                    in al,dx
                    and al,00010000B        ; tes CTS
                    jnz TES_THRE
                    loop TES_CTS
;
                    LF_CR
                    tulis CTS_high          ; CTS = 0 berarti
                                            ;pin CTS high
                    LF_CR
;
TES_THRE:           mov cx,OFFFFH
CHECK_THRE:         mov dx,LINE_STAT_REG
                    in al,dx
                    and al,00100000B
                    jnz TES_TSRE
                    loop CHECK_THRE
                    TULIS tul_TX_no_empty
                    LF_CR
                    LF_CR
;
TES_TSRE:           mov cx,OFFFFH
CHECK_TSRE:         mov dx,LINE_STAT_REG
                    in al,dx
                    and al,01000000B
                    jnz BACK
                    loop CHECK_TSRE
                    TULIS tul_TX_no_empty
                    LF_CR
                    LF_CR
BACK:
                    xor ax,ax
                    ret
Handshake           endp
;-----Procedure DATA OUT_1-----
Data_out_1          Proc Near
```

```
COBA_KIRIM_HURUF:
                        Call Handshake
                        mov al,simpanan_huruf
                        inc al
                        cmp al,05Bh
                        je KIRIM_KEY_IN

                        mov simpanan_huruf,AL
                        mov dx,TX_BUFFER
                        out dx,al
                        jmp COBA_KIRIM_HURUF
KIRIM_KEY_IN:
                        ret
Data_out_1            endp
;------Procedure Counter-------------
Line_col_counter   Proc Near
                        push ax
                        push dx
                        mov dl,column
                        mov dh,row
                        cmp dh,9
                        jle RIN1
                        Call Two_digit
                        jmp KUOLOM
RIN1:
                        add dh,030h
                        push dx
                        mov al,' '
                        mov bl,color
                        gotoyx bar,kol1,hal
                        tul1karw
                        inc kol1
                        gotoyx bar,kol1,hal
                        pop dx
                        mov al,dh
                        tul1karw
KUOLOM:
                        cmp dl,9
                        jle RIN2
                        Call Dua_digit
                        jmp   ERIN
RIN2:
                        add dl,030h
                        push dx
                        mov al,' '
                        mov bl,color
                        gotoyx bar,kol2,hal
                        tul1karw
                        inc kol2
                        gotoyx bar,kol2,hal
                        pop dx
                        mov al,dl
                        tul1karw
ERIN:
                        pop dx
                        pop ax
```

```
                        ret
Line_col_counter   endp
;---------Procedure Two Digit---------
Two_digit           Proc Near
                       push ax
                       push dx
                       push dx
                       cmp dh,19
                       ja NGURANG
                       sub dh,10
                       jmp NAMBAH
NGURANG:
                       sub dh,20
NAMBAH:
                       add dh,030h
                       push dx
                       inc kol1
                       gotoyx bar,kol1,hal
                       pop dx
                       mov al,dh
                       mov bl,color
                       tul1karw
                       pop dx
                       cmp dh,19
                       jle SATU
                       mov al,032h
                       jmp TEMPAT
SATU:
                       mov al,031h
TEMPAT:
                       dec kol1
                       gotoyx bar,kol1,hal
                       tul1karw
                       pop dx
                       pop ax
                       ret
Two_digit           endp
;---------Procedure Dua Digit--------
Dua_digit           Proc Near
                       push ax
                       push dx
                       push dx
                       cmp dl,78
                       ja KURANGI1
                       cmp dl,69
                       ja KURANGI2
                       cmp dl,59
                       ja KURANGI3
                       cmp dl,49
                       ja KURANGI4
                       cmp dl,39
                       ja KURANGI5
                       cmp dl,29
                       ja KURANGI6
                       cmp dl,19
                       ja KURANGI7
```

```
                sub dl,10
                jmp TAMBAH
KURANGI1:
                sub dl,80
                jmp TAMBAH
KURANGI2:
                sub dl,70
                jmp TAMBAH
KURANGI3:
                sub dl,60
                jmp TAMBAH
KURANGI4:
                sub dl,50
                jmp TAMBAH
KURANGI5:
                sub dl,40
                jmp TAMBAH
KURANGI6:
                sub dl,30
                jmp TAMBAH
KURANGI7:
                sub dl,20
TAMBAH:
                add dl,030h
                push dx
                inc kol2
                gotoyx bar,kol2,hal
                pop dx
                mov al,dl
                mov bl,color
                tul1karv
                pop dx
                cmp dl,78
                jle SEVEN
                mov al,038h
                jmp WADAH
SEVEN:
                cmp dl,69
                jle SIX
                mov al,037h
                jmp WADAH
SIX:
                cmp dl,59
                jle FIVE
                mov al,036h
                jmp WADAH
FIVE:
                cmp dl,49
                jle FOUR
                mov al,035h
                jmp WADAH
FOUR:
                cmp dl,39
                jle THREE
                mov al,034h
                jmp WADAH
```

```
THREE:
                        cmp dl,29
                        jle TWO
                        mov al,033h
                        jmp WADAH
TWO:
                        cmp dl,19
                        jle ONE
                        mov al,032h
                        jmp WADAH
ONE:
                        mov al,031h

WADAH:
                        dec kol2
                        gotoyx bar,kol2,hal
                        tul1karw
                        pop dx
                        pop ax
                        ret
Dua_digit             endp
;-----Procedure Check Space TX------
Check_space_TX     Proc Near
                      cmp   al,' '
                      jne OK
                      Call Enter_TX
OK:
                      ret
Check_space_TX     endp
;-----Procedure Check Space RX------
Check_space_RX     Proc Near
                      cmp   al,' '
                      jne OK2
                      Call Enter_RX
OK2:
                      ret
Check_space_RX     endp
;--------Procedure Enter TX---------
Enter_TX              Proc Near
                       mov CR,1
                       dec column_k
                       mov dl,column_k
                       mov dh,row_k
                       Call Simpan_koor_kur
                       mov column_k,1
                       inc row_k
                       cmp row_k,12
                       jnz PLACE_KUR
                       mov row_k,11
PLACE_KUR:
                       gotoyx row_k,column_k,page_k
                       ret
Enter_TX              endp
;---------Procedure Enter RX---------
Enter_RX              Proc Near
                       mov al,' '
```

```
                    dec column_t
                    mov dl,column_t
                    mov dh,row_t
                    Call Simpan_koor_kurRX
                    mov column_t,1
                    inc row_t
                    cmp row_t,23
                    jnz PASANG
                    mov row_t,22
PASANG:             ret
Enter_RX            endp
;----------Procedure Clear Screen---------
clear_screen        proc near
                    mov ax,0b800h
                    mov es,ax
                    mov di,address_offset
                    mov bl,' '
                    mov bh,0
                    mov cx,cacah_huruf
                    cld

                    mov dx,03dah
tunggu_0:           in al,dx
                    test al,1
                    jnz tunggu_0

tunggu_1:           in al,dx
                    test al,1
                    jnz tunggu_1

                    mov ax,bx
                    stosw
                    loop tunggu_0
                    ret
clear_screen        endp
;----------Procedure Bersih---------
Bersih              Proc Near
                    mov ch,BARATAS
                    mov cl,1
                    mov dh,BARBAW
                    mov dl,78
                    mov bh,WARNA
                    mov al,0

                    mov ah,06h
                    int 10h
                    ret
Bersih              endp
;--Procedure Simpan Koord Kursor TX--
Simpan_koor_kur     Proc Near
                    cmp dh,1
                    jne BARIS_2
                    mov KOORDY_BAR1,dl
                    jmp ASAL
BARIS_2:
                    cmp dh,2
```

```
                        jne BARIS_3
                        mov KOORDY_BAR2,dl
                        jmp ASAL
BARIS_3:
                        cmp dh,3
                        jne BARIS_4
                        mov KOORDY_BAR3,dl
                        jmp ASAL
BARIS_4:
                        cmp dh,4
                        jne BARIS_5
                        mov KOORDY_BAR4,dl
                        jmp ASAL
BARIS_5:
                        cmp dh,5
                        jne BARIS_6
                        mov KOORDY_BAR5,dl
                        jmp ASAL
BARIS_6:
                        cmp dh,6
                        jne BARIS_7
                        mov KOORDY_BAR6,dl
                        jmp ASAL
BARIS_7:
                        cmp dh,7
                        jne BARIS_8
                        mov KOORDY_BAR7,dl
                        jmp ASAL
BARIS_8:
                        cmp dh,8
                        jne BARIS_9
                        mov KOORDY_BAR8,dl
                        jmp ASAL
BARIS_9:
                        cmp dh,9
                        jne BARIS_10
                        mov KOORDY_BAR9,dl
                        jmp ASAL
BARIS_10:
                        cmp dh,10
                        jne BARIS_11
                        mov KOORDY_BAR10,dl
                        jmp ASAL
BARIS_11:
                        mov KOORDY_BAR11,dl
ASAL:                  ret
Simpan_koor_kur   endp
;--Procedure Simpan Koord Kursor RX--
Simpan_koor_kurRX   Proc Near
                        cmp dh,13
                        jne BARIS_14
                        mov KOORDY_BAR13,dl
                        jmp ASAL_RX
BARIS_14:
                        cmp dh,14
                        jne BARIS_15
```

```
                    mov KOORDY_BAR14,dl
                    jmp ASAL_RX
BARIS_15:
                    cmp dh,15
                    jne BARIS_16
                    mov KOORDY_BAR15,dl
                    jmp ASAL_RX
BARIS_16:
                    cmp dh,16
                    jne BARIS_17
                    mov KOORDY_BAR16,dl
                    jmp ASAL_RX
BARIS_17:
                    cmp dh,17
                    jne BARIS_18
                    mov KOORDY_BAR17,dl
                    jmp ASAL_RX
BARIS_18:
                    cmp dh,18
                    jne BARIS_19
                    mov KOORDY_BAR18,dl
                    jmp ASAL_RX
BARIS_19:
                    cmp dh,19
                    jne BARIS_20
                    mov KOORDY_BAR19,dl
                    jmp ASAL_RX
BARIS_20:
                    cmp dh,20
                    jne BARIS_21
                    mov KOORDY_BAR20,dl
                    jmp ASAL_RX
BARIS_21:
                    cmp dh,21
                    jne BARIS_22
                    mov KOORDY_BAR21,dl
                    jmp ASAL_RX
BARIS_22:
                    cmp dh,22
                    jne BARIS_23
                    mov KOORDY_BAR22,dl
                    jmp ASAL_RX
BARIS_23:
                    mov KOORDY_BAR23,dl
ASAL_RX:           ret
Simpan_koor_kurRX   endp
;--Procedure Compare Koord Baris TX--
Compare_pos_bar   Proc Near
                    cmp dh,1
                    jne BAR_2
                    mov dl,KOORDY_BAR1
                    jmp kmon
BAR_2:
                    cmp dh,2
                    jne BAR_3
                    mov dl,KOORDY_BAR2
```

```
                        jmp  kmon
BAR_3:
                        cmp  dh,3
                        jne  BAR_4
                        mov  dl,KOORDY_BAR3
                        jmp  kmon
BAR_4:
                        cmp  dh,4
                        jne  BAR_5
                        mov  dl,KOORDY_BAR4
                        jmp  kmon
BAR_5:
                        cmp  dh,5
                        jne  BAR_6
                        mov  dl,KOORDY_BAR5
                        jmp  kmon
BAR_6:
                        cmp  dh,6
                        jne  BAR_7
                        mov  dl,KOORDY_BAR6
                        jmp  kmon
BAR_7:
                        cmp  dh,7
                        jne  BAR_8
                        mov  dl,KOORDY_BAR7
                        jmp  kmon
BAR_8:
                        cmp  dh,8
                        jne  BAR_9
                        mov  dl,KOORDY_BAR8
                        jmp  kmon
BAR_9:
                        cmp  dh,9
                        jne  BAR_10
                        mov  dl,KOORDY_BAR9
                        jmp  kmon
BAR_10:
                        cmp  dh,10
                        jne  BAR_11
                        mov  dl,KOORDY_BAR10
                        jmp  kmon
BAR_11:
                        mov  dl,KOORDY_BAR11
kmon:                  ret
Compare_pos_bar   endp
;--Procedure Compare Koord Baris Rx--
Compare_pos_barRX   Proc Near
                        cmp  dh,13
                        jne  BAR_14
                        mov  dl,KOORDY_BAR13
                        jmp  kmon_RX
BAR_14:
                        cmp  dh,14
                        jne  BAR_15
                        mov  dl,KOORDY_BAR14
                        jmp  kmon_RX
```

```
BAR_15:
                    cmp dh,15
                    jne BAR_16
                    mov dl,KOORDY_BAR15
                    jmp kmon_RX
BAR_16:
                    cmp dh,16
                    jne BAR_17
                    mov dl,KOORDY_BAR16
                    jmp kmon_RX
BAR_17:
                    cmp dh,17
                    jne BAR_18
                    mov dl,KOORDY_BAR17
                    jmp kmon_RX
BAR_18:
                    cmp dh,18
                    jne BAR_19
                    mov dl,KOORDY_BAR18
                    jmp kmon_RX
BAR_19:
                    cmp dh,19
                    jne BAR_20
                    mov dl,KOORDY_BAR19
                    jmp kmon_RX
BAR_20:
                    cmp dh,20
                    jne BAR_21
                    mov dl,KOORDY_BAR20
                    jmp kmon_RX
BAR_21:
                    cmp dh,21
                    jne BAR_22
                    mov dl,KOORDY_BAR21
                    jmp kmon_RX
BAR_22:
                    cmp dh,22
                    jne BAR_23
                    mov dl,KOORDY_BAR22
                    jmp kmon_RX
BAR_23:
                    mov dl,KOORDY_BAR23
kmon_RX:          ret
Compare_pos_barRX   endp
;------Procedure Kursor Maju TX------
Kursor_maju_TX    PROC   NEAR
                    mov dl,column_k
                    mov dh,row_k
                    mov bh,page_k
                    inc dl
                    cmp dl,78
                    jbe TARUH_KURSOR
                    mov dl,78
                    Call Simpan_koor_kur
                    mov dl,1
                    inc dh
```

```
                        cmp dh,12
                        je SCROLL_KE_ATAS
;
TARUH_KURSOR:       mov bh,0
                        mov ah,2
                        int 10h
                        mov CLEAR,0
                        ret
SCROLL_KE_ATAS:
                        mov CLEAR,1
                        RET
Kursor_maju_TX     ENDP
;-----Procedure Kursor Maju RX-----
kursor_maju_RX     PROC   NEAR
                        mov dl,column_t
                        mov dh,row_t
                        mov bh,page_t
                        inc dl
                        cmp dl,78
                        jbe TARUH_KURSOR2
                        mov dl,78
                        Call Simpan_koor_kurRX
                        mov dl,1
                        inc dh
                        cmp dh,23
                        je SCROLL_KE_ATAS2
;
TARUH_KURSOR2:     mov bh,0
                        mov ah,2
                        int 10h
                        mov CLEAR,0
                        ret
SCROLL_KE_ATAS2:
                        mov CLEAR,1
                        ret
kursor_maju_RX     ENDP
;----Procedure Kursor Maju------
kursor_maju         PROC   NEAR
                        MOV BH,0
                        MOV AH,03H
                        INT 10H
                        INC DL
                        CMP DL,78
                        JBE taruh_kursor3
                        MOV DL,0
                        INC DH
                        CMP DH,11
                        JA scroll_ke_atas3
;
taruh_kursor3:     MOV BH,0
                        MOV AH,2
                        INT 10H
                        MOV CLEAR,0
                        RET
scroll_ke_atas3:
                        MOV CLEAR,1
```

```
                    RET
kursor_maju         ENDP
;-----Procedure Kursor Mundur TX-----
Kursor_mundur_TX    PROC NEAR
                    DEC COLUMN_K
                    CMP COLUMN_K,1
                    JAE TERUS
                    cmp row_k,3
                    jle PANCET
                    MOV COLUMN_K,78
                    DEC ROW_K
TERUS:
                    CMP ROW_K,3
                    JA ATUR_KUR
PANCET:
                    CMP COLUMN_K,1
                    JA ATUR_KUR
                    MOV ROW_K,3
                    MOV COLUMN_K,1
                    MOV PAGE_K,0
ATUR_KUR:           ret
Kursor_mundur_TX    endp
;-----Procedure Kursor Mundur RX-----
Kursor_mundur_RX    PROC NEAR
                    DEC COLUMN_T
                    CMP COLUMN_T,1
                    JAE CONTINUE
                    CMP ROW_T,14
                    JLE STEADY
                    MOV COLUMN_T,78
                    DEC ROW_T
CONTINUE:
                    CMP ROW_T,14
                    JA TATA
STEADY:
                    CMP COLUMN_T,1
                    JA TATA
                    MOV ROW_T,14
                    MOV COLUMN_T,1
                    MOV PAGE_T,0
TATA:               RET
Kursor_mundur_RX    ENDP
;
;Subroutine menulis 1 baris huruf yang dibatasi $
;-----------------------------------------------
tulis_berwarna      PROC  NEAR
                    MOV BYTE PTR [simpanan_isi_BL],BL
huruf_berikut:      MOV BL,BYTE PTR [simpanan_isi_BL]
                    MOV AL,BYTE PTR [SI]
                    CMP AL,'$'
                    JE terus_balik
;
                    INC SI
;
                    tulikarv
                    CALL kursor_maju
```

```
;
                        JMP huruf_berikut
terus_balik:          RET
tulis_berwarna        ENDP

;---------Procedure Display-----------
Disply      Proc Near
              gotoyx 80,25,0
              @parstr 0,0,56,80,<offset data1>
              @parstr 24,0,56,80,<offset data2>
              @parwin 0,79,1,23,31,2,0,31
              @parstr 1,16,31,48,<offset data3>
              @parstr 2,1,0eah,24,<offset data_kirim>
              @parstr 2,45,06ah,21,<offset data_line_column>
              @parvin 0,79,12,23,31,2,0,31
              @parstr 12,0,31,80,<offset batas>
              @parstr 13,1,0eah,24,<offset data_terima>
              @parstr 13,45,06ah,21,<offset data_line_column>
              @parvin 18,61,19,22,30,3,0,0
              @parstr 20,19,30,42,<offset Pre4>
              @parstr 21,19,30,42,<offset Pre5>
              ret
Disply      endp

;----------Procedure setting kursor---------
Setting_kur          Proc Near
                        mov row_k,3
                        mov column_k,1
                        mov page_k,0
                        mov row_t,14
                        mov column_t,1
                        mov page_t,0
                        ret
Setting_kur          endp

;----------Procedure Blok String------------
Block                 Proc Near
                        mov BARATAS,3
                        mov BARBAW,11
                        mov WARNA,01eh
                        Call Bersih
                        mov row_k,3
                        mov column_k,0
                        mov page_k,0
                        @parstr 0,0,56,80,<offset data1>
                        mov al,01h     ;tombol F1 ditekan untuk CLS
                        jmp LHOE
LHOE:
                        ret
Block                 endp

;---------Procedure Print Screen-----------
PrtScr                Proc Near
                        int 5h
                        jmp LHAE
LHAE:
```

```
                    @parstr 0,0,56,80,<offset data1>
                    ret
PrtScr              endp

;------Procedure Print---------------
Print               Proc Near
                    @parget 40,64,1,5,<offset offmen5>
                    @parget 42,64,2,5,<offset offmen6>
                    @parput 42,64,2,5,7,<offset offmen6>
                    @parwin 40,62,1,4,40,1,0,31
                    @parstr 2,41,47,21,<offset writefl>
                    @parstr 3,41,121,21,<offset newfill>
                    @askoo
                    cmp al,49
                    jne DUAA
                    mov BPS_MS,09h
                    mov BPS_LS,0h
                    jmp CABUT1a
DUAA:               cmp al,50
                    jne TIGAA
                    mov BPS_MS,06h
                    mov BPS_LS,0h
                    jmp CABUT1a
TIGAA:              cmp al,51
                    jne EMPATT
                    mov BPS_MS,04h
                    mov BPS_LS,017h
                    jmp CABUT1a
EMPATT:             cmp al,52
                    jne LIMAA
                    mov BPS_MS,03h
                    mov BPS_LS,059h
CABUT1a:            jmp CABUT1
LIMAA:              cmp al,53
                    jne ENAMM
                    mov BPS_MS,03h
                    mov BPS_LS,0h
                    jmp CABUT1
ENAMM:              cmp al,54
                    jne CABUT
                    mov BPS_MS,01h
                    mov BPS_LS,080h
CABUT1:
                    Call INITIATE
                    Call Handshake
CABUT:
                    @parput 40,64,1,5,0,<offset offmen5>
                    @parstr 0,0,56,80,<offset data1>
                    ret
Print               endp

;-----------Procedure Window------------
Window               Proc Near
                     push ax
                     push bx
                     push cx
```

```
                        push dx
;pojok kiri atas
                        mov dh,barata
                        mov dl,kolata
                        mov al,jenis
                        cmp al,4
                        jg kela
                        lea bx,L_corner
                        dec al
                        xlat L_corner
kela:
                        mov ah,att
                        mov bx,1
                        mov cx,1
                        Call Oxxem              ;dx,cx,bx
;pojok kanan atas
                        mov dl,kolbaw
                        mov al,jenis
                        cmp al,4
                        jg kelaa
                        lea bx,R_corner
                        dec al
                        xlat R_corner
kelaa:
                        mov ah,att
                        mov bx,1
                        mov cx,1
                        Call Oxxem
;pojok kanan bawah
                        mov dh,barbawa
                        mov dl,kolbaw
                        mov al,jenis
                        cmp al,4
                        jg kelab
                        lea bx,BR_corner
                        dec al
                        xlat BR_corner
kelab:
                        mov ah,att
                        mov bx,1
                        mov cx,1
                        Call Oxxem              ;dx,cx,bx
;pojok kiri bawah
                        mov dh,barbawa
                        mov dl,kolata
                        mov al,jenis
                        cmp al,4
                        jg kelac
                        lea bx,BL_corner
                        dec al
                        xlat BL_corner
kelac:
                        mov ah,att
                        mov bx,1
                        mov cx,1
                        Call Oxxem
```

```
;baris atas bawah
                        mov dh,barata
                        mov dl,kolata
                        add dl,1
                        mov cx,1
                        mov al,jenis
                        cmp al,4
                        jg kelad
                        lea bx,horiz
                        dec al
                        xlat horiz
kelad:
                        mov ah,att
                        mov bl,kolbaw
                        sub bl,dl
                        xor bh,bh
                        Call Oxxem              ;dx,cx,bx
                        push ax
                        push bx
                        mov dh,barbawa
                        pop bx
                        pop ax
                        Call Oxxem
;kolom kanan kiri
                        mov dh,barata
                        mov dl,kolata
                        inc dh
                        mov cl,barbawa
                        sub cl,dh
                        xor ch,ch
                        mov al,jenis
                        cmp al,4
                        jg kelae
                        lea bx,Verti
                        dec al
                        xlat Verti
kelae:
                        mov ah,att
                        mov bx,1
                        Call Oxxem
                        push ax
                        push bx
                        mov dl,kolbaw
                        pop bx
                        pop ax
                        call Oxxem
;isi perut window
                        mov dh,barata
                        inc dh
                        mov dl,kolata
                        inc dl
                        mov cl,barbawa
                        sub cl,dh
                        xor ch,ch
                        mov bl,kolbaw
                        sub bl,dl
```

```
                        xor bh,bh
                        mov al,charfil
                        mov ah,attfil
                        call Oxxem
                        pop dx
                        pop cx
                        pop bx
                        pop ax
                        ret
Windov                 endp

;------Procedure Menulis Karakter di Display---------;
Oxxem                   Proc Near
                        push bx
                        push cx
                        push si
                        push di
                        push es
                        cmp cx,1
                        jl sel
                        cmp bx,1
                        jl sel
                        push ax
                        push dx
                        mov ax,80
                        mul dh
                        xor dh,dh
                        mov si,dx
                        add si,ax
                        shl si,1
                        pop dx
                        pop cx
                        mov di,0B800h
                        mov es,di
lopfil:
                        push cx
                        mov cx,bx
                        mov di,si
                        cld
fillscr:
                        stosw
                        loop fillscr
                        add si,160
                        pop cx
                        loop lopfil
sel:
                        pop es
                        pop di
                        pop si
                        pop cx
                        pop bx
                        ret
Oxxem                  endp

;-----Procedure Menulis String pada layar---------
Putstr                 Proc Near
```

```
                        push ax
                        push bx
                        push cx
                        push si
                        push di
                        mov bx,baris
                        mov ax,80
                        mul bl
                        mov di,kolom
                        add di,ax
                        shl di,1

                        mov cx,jumstr
                        cmp cx,1
                        jl sldr
                        cmp cx,3E80h
                        jg sldr
                        mov bx,attrib
                        mov ah,bl
                        mov si,string
                        mov bx,0B800h
                        mov es,bx
more:                   mov al,byte ptr[si]
                        mov es:[di],al
                        inc di
                        mov es:[di],ah
                        inc di
                        inc si
                        loop more
sldr:
                        pop di
                        pop si
                        pop cx
                        pop bx
                        pop ax
                        ret
Putstr                  endp

;--------Procedure Getimmage-----------------
Getimmage               Proc Near
                        push ax
                        push bx
                        push cx
                        push dx
                        push si
                        push di
                        push ds
                        push es
                        mov cl,kolata
                        mov bl,kolbaw
                        cmp bl,cl
                        jl error
                        sub bl,cl
                        inc bl
                        xor dx,dx
                        add dl,bl
```

```
                        shl dx,1
                        mov al,barata
                        mov bl,barbawa
                        cmp bl,al
                        jl error
                        sub bl,al
                        inc bl
                        mov ah,cl
                        xor cx,cx
                        add cl,bl
                        push cx
                        mov bx,ax
                        mov al,160
                        mul bl
                        mov cx,ax
                        xor ax,ax
                        mov al,bh
                        shl ax,1
                        add cx,ax
                        mov bx,cx
                        pop cx
                        mov ax,@data    ;segment
                        mov es,ax
                        mov ax,offmd    ;offset
                        mov di,ax
                        mov si,0B800h
                        mov ds,si
JAUR:                   push cx
                        mov si,bx
                        mov cx,dx
                        cld
KAUR:                   movsb
                        loop KAUR
                        add bx,160
                        pop cx
                        loop JAUR
ERROR:
                        pop es
                        pop ds
                        pop di
                        pop si
                        pop dx
                        pop cx
                        pop bx
                        pop ax
                        ret
Getimmage               endp

;-------------Procedure Putimmage-----------------
Putimmage               Proc Near
                        push ax
                        push bx
                        push cx
                        push dx
                        push si
                        push di
```

```
                push ds
                push es
                mov cl,kolata
                mov bl,kolbaw
                cmp bl,cl
                jl ERRO
                sub bl,cl
                inc bl
                xor dx,dx
                add dl,bl
                mov al,barata
                mov bl,barbava
                cmp bl,al
                jl ERRO
                sub bl,al
                inc bl
                mov ah,cl
                xor cx,cx
                add cl,bl
                push cx
                mov bx,ax
                mov al,100
                mul bl
                mov cx,ax
                xor ax,ax
                mov al,bh
                shl ax,1
                add cx,ax
                mov bx,cx
                push bx
                jmp LOMPAT
ERRO:           jmp ERROSS
LOMPAT:
                mov ax,atchok
                mov si,offme
                mov bx,sdata
                mov ds,bx
                mov di,0B900h
                mov es,di
                pop bx
                pop cx
                cmp al,0
                je NOCHANGE
                mov ah,al
XXJAUR:
                push cx
                mov di,bx
                mov cx,dx
XXKAUR:
                mov al,ds:[si]
                mov es:[di],al
                inc si
                inc di
                mov es:[di],ah
                inc si
                inc di
```

```
                        loop XXKAUR
                        add bx,160
                        pop cx
                        loop XXJAUR
                        jmp ERRORR
NOCHANGE:
XJAUR:                  push cx
                        mov di,bx
                        mov cx,dx
                        shl cx,1
                        cld
XKAUR:                  movsb
                        loop XKAUR
                        add bx,160
                        pop cx
                        loop XJAUR
ERRORR:                 pop es
                        pop ds
                        pop di
                        pop si
                        pop dx
                        pop cx
                        pop bx
                        pop ax
                        ret
PutImage                endp

;
;-----------End Of Procedure-----------------
                        END
```

07 DEC 1992

FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO - ITS

TUGAS AKHIR (EE 1799)

Nama Mahasiswa	:	EDWIN ARISTIAWAN
Nomor Pokok	:	2882200941
Bidang Studi	:	Elektronika
Tugas diberikan	:	Akhir semester ganjil 1992/1993
Tugas diselesaikan	:	Akhir semester genap 1992/1993
Dosen Pembimbing	:	1. Ir. Moch. Moefadol Asyari 2. Ir. Hendra Kusuma
JUDUL TUGAS AKHIR	:	ALAT PENGIRIM DATA DIGITAL MELALUI PEMANCAR RADIO FM STEREO YANG DAPAT DIINTERFACEKAN DENGAN IBM-PC

URAIAN TUGAS AKHIR :

Perkembangan Teknologi Elektronika Telekomunikasi yang sedemikian cepat, memungkinkan penggunaan saluran telekomunikasi yang semakin luas, khususnya saluran radio. Pada saat ini banyak sekali stasiun pemancar radio FM yang bermunculan di kota-kota besar di Indonesia. Sistem pemancar ini mulai disukai karena dapat menghasilkan suara yang jernih dan efek stereo pada pesawat radio penerima. Alat ini dirancang untuk mengoptimalkan fungsi dari pemancar FM tersebut. Dengan alat ini dapat dikirimkan data digital melalui pemancar radio FM. Sedangkan pada penerimanya dapat digunakan radio penerima FM biasa yang sudah ditambah dengan peralatan yang dapat mereproduksi data digital pada display sesuai dengan data yang dikirimkan pada pemancar.

Menyetujui

Koordinator Bidang Studi
Elektronika

(Ir. Soetikno)
Nip. 130445231

Dosen Pembimbing I

(Ir. Moch. Moefadol Asyari)
Nip. 130422814

Dosen Pembimbing II

(Ir. Hendra Kusuma)
Nip. 131846104

Mengetahui
Ketua Jurusan Teknik Elektro
FTI - ITS

(Ir. Katjuk Astrowulan, MSEE)
Nip. 130687438

USULAN TUGAS AKHIR

1. JUDUL : ALAT PENGIRIM DATA DIGITAL MELALUI PEMANCAR RADIO FM-STEREO YANG DAPAT DIINTERFACEKAN DENGAN IBM - PC

2. Bidang Studi : Elektronika

3. Ruang Lingkup :
   - Dasar Sistem Komunikasi
   - Mikroelektronika
   - Mikroprosessor
   - Instrumentasi Elektronika
   - Bahasa Assembly

4. Latar Belakang : Perkembangan Teknologi Elektro - nika Telekomunikasi yang sedemikian cepat, memungkinkan penggunaan saluran telekomunikasi yang semakin luas, khususnya saluran radio. Pada saat ini banyak sekali stasiun pemancar radio FM di kota-kota besar di Indonesia. Sistem pemancar ini mulai disukai karena dapat menghasilkan suara yang jernih dan efek stereo pada pesawat radio penerima FM. Dengan alat

ini dapat dikirimkan data digital melalui pemancar radio FM. Sedangkan pada penerimanya dapat digunakan radio penerima FM biasa yang sudah ditambah dengan peralatan yang dapat mereproduksi data digital pada display sesuai dengan data yang dikirim pada pemancar. Bila dikembangkan lebih lanjut peralatan ini mempunyai fungsi khusus dan umum. Fungsi khusus adalah digunakan untuk kepentingan pemerintahan daerah atau instansi tertentu, misalnya: untuk penyampaian instruksi yang bersifat rahasia (Radiopaging). Sedangkan untuk fungsi umum misalnya untuk menampilkan identitas dari stasiun pemancar radio FM yang sedang didengarkan.

5. Penelaahan Studi :

- Mempelajari cara kerja mikro - prosessor IBM-PC dan teknik interfacingnya.
- Mempelajari cara kerja pemancar dan penerima radio FM Stereo
- Mempelajari sistem pengolahan data.

- Mempelajari bahasa pemrograman komputer.

6. Tujuan :
   - Merencanakan dan membuat alat yang dapat menginterfacekan IBM-PC dengan pemancar radio FM Stereo dalam fungsinya sebagai pengirim data digital
   - Merencanakan dan membuat alat yang dapat menampilkan kembali data digital pada radio penerima.
   - Merencanakan dan membuat perangkat lunak untuk mendukung perangkat keras dalam fungsinya sebagai alat pengirim data digital melalui pemancar radio FM Stereo.

7. Langkah - langkah :
   - Studi Literatur
   - Perencanaan alat
   - Pembuatan alat
   - Pengukuran dan pengujian alat
   - Penyusunan naskah.

8. Relevansi : Alat ini diharapkan dapat berguna untuk lebih mengoptimalkan fungsi pemancar radio FM stereo.

9. Jadwal Kegiatan : Seluruh kegiatan diharapkan dapat diselesaikan dalam waktu enam bulan dengan jadwal sebagai berikut :

KEGIATAN	BULAN KE					
	1	2	3	4	5	6
Studi Literatur						
Perencanaan alat						
Pembuatan alat						
Pengujian alat						
Penyusunan naskah						